THE TIMEX-SINCLAIR 1000

IDEA BOOK

INCLUDES 50 READY-TO-RUN EDUCATIONAL PROGRAMS

DAVID H. AHL

PROGRAM CONVERSIONS BY DAVID AND JAMES GROSJEAN

The Timex Sinclair 1000 Ideabook

NOT FOR RESALE

The Timex Sinclair 1000 Ideabook

Includes 50 Ready-to-Run Programs

David H. Ahl

Program Conversions:
David and James Grosjean

Illustrations: Wayne Kaneshiro

Creative Computing Press
Morris Plains, New Jersey

Copyright ©1983 by Creative Computing

All rights reserved. No portion of this book may be reproduced—mechanically, electronically or by any other means, including photocopying—without permission of the publisher.

Library of Congress Number 83-70231
ISBN 0-916688-48-8

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Creative Computing Press
39 E. Hanover Avenue
Morris Plains, NJ 07950

Dedication: To Kato, who was no help at all in getting this book written.

About the Author

David H. Ahl has a BEE from Cornell University, MBA from Carnegie-Mellon University and has done further work in educational psychology at the University of Pittsburgh.

He served in the Army Security Agency, was a consultant with Management Science Associates and a senior research fellow with Educational Systems Research Institute.

In early 1970, he joined Digital Equipment Corporation. As education product line manager, he formulated the concept of an educational computer system consisting of hardware, software and courseware and helped guide DEC into a leading position in the education market.

Mr. Ahl joined AT&T in 1974 as education marketing manager and was later promoted to manager of marketing communications for the unit later to become American Bell. Concurrent with this move, he started *Creative Computing* as a hobby in late 1974.

As *Creative Computing* grew, Mr. Ahl left AT&T in 1978 to devote full time to it. *Creative Computing* magazine today is Number 1 in software and applications. In January 1980, Ahl founded *SYNC* magazine and, over the years, has acquired or started several other publications.

Mr. Ahl is the author or editor of 16 books and over 150 articles about the use of computers. He is a frequent lecturer and workshop leader at educational and professional conferences.

Contents

Preface	IX
Program Conversion Notes	2
1. Drill and Practice	11
Addition practice	12
Addition practice, adjusted by grade level	14
Time/speed/distance problems	18
Kinematics problems	20
2. Problem Solving	23
How many tickets? (Flow Chart)	24
Drinking and blood pressure	25
Two simultaneous equations	28
Quadratic equation solver	30
Exponential equation solver	32
Roots of any function	34
Plot any function	36
3. Sets and Repetitive Trials	38
Group of girls and boys	39
Brown's books	41
Intersection of sets	43
Prime factors	45
Greatest common divisor	47
Cryptarithmetic problem	49
Sailors and monkey	53
Palindromes	56
4. Convergence and Recursion	59
Change for a dollar	60
Change for any amount to $5.00	62
Converge on e and pi	65
Converge on pi revisited	68
Length of any curve	70
Converge on a square root	72

5. Compounding 75
Indians and interest 76
Systematic savings 78
Systematic savings revisited 80
Loan payments 82
Interest on credit purchases 85
Population growth 87

6. Probability 89
Pascal's triangle—calculated 90
Pascal's triangle—by probability 92
Common birthdays 94
Coins in a pocket 96
Baseball cards 98
System reliability 101

7. Geometry and the Calculus 103
Crossed and slipped ladders 104
Distance between coordinate points 107
Area—by calculation 109
Area—by integration 111

8. Science 114
Gas Volumes 115
Charles' Law drill 117
Boyle's Law drill 120
Photoelectric emissions 123
Mutation of moths 126

9. Potpourri 129
Number guessing game 130
Depreciation—three methods 132
Smog simulation 134
Lunar landing simulation 137

References 141

Preface

For the past 20 years, Clive Sinclair has been fascinated with small electronic devices. Over the years, he has set up one mail order business after another. In the early days, he sold transistors, then miniature radios. In the early 1970's, he designed one of the earliest pocket calculators with advanced mathematical functions. His "Black Watch" kit was one of the first inexpensive digital watches.

In 1980, Sinclair introduced the world's smallest and cheapest personal computer, the Sinclair ZX80. In contrast to previous successes, which had been modest, the ZX80 took the British personal computer market by storm. Few other computers were made in England, and thus, because of high import duties, foreign-built units were considerably more expensive than the diminutive, made-in-Scotland ZX80. In mid-1981, the ZX80 was introduced into other markets around the world, including the U.S.

Improving upon the design of the ZX80, the ZX81 was introduced in late 1981. Shortly thereafter, Sinclair made a deal with W.H. Smith, a large British retailer of books, cards, and stationery, to handle the ZX81. Following that, another deal was hammered out with Timex to sell the ZX81 in the U.S. market. For retail sales, the computer was renamed the Timex Sinclair 1000, and 2K of memory installed instead of the original 1K. Within a matter of months, sales started to soar.

Although small, the Timex Sinclair 1000 is just as much a computer as the Apple, Atari, or IBM PC. However, because of the small memory, the machine cannot store a large data base, do word processing, or display color graphics. Also, problems that could be solved by writing one large program on another computer, have to be broken down into smaller pieces for the Sinclair. With the Sinclair computer, it is more important to think through a problem completely and determine what can be done "off line" and what must be done on the computer.

That is what this book is all about. While it has more than 50 ready-to-run programs, the main thing you should look for from the book is an approach to solving problems—big and small. Some of the problems demonstrate the capability of the computer; others identify its shortcomings. It is important to be familiar with both the strengths and weaknesses of your tools so you can recognize the types of jobs for which they are suitable (and not suitable).

This book, as its title suggests, focuses primarily on educational applications for the computer. There are many other excellent sources of information about other applications and for making best use of the Sinclair. One of the best is the magazine, *SYNC,* which contains a wide variety of information about software, applications, peripherals, and even hardware modifications.

This book is designed to be read with a working computer at hand. While there is textual material to be read, the most important things are the experiments and problems to be tried with your own computer. The book raises many questions for which you should try to find answers. There are no answers to these questions and problems in the back of the book; you should be able to discover the answers as you work the problems out on your computer.

You will be able to incorporate many of the routines and approaches in the book into programs of your own as you use your computer to deal with "real world" problems. Other programs will simply point you in the right direction. And some of the programs in the book are just plain fun. Learn. Experiment. Have fun!

Morris Plains, New Jersey
May 1983

David H. Ahl

The Timex Sinclair 1000 Ideabook

Program Conversion Notes

The programs in this book were originally written over a ten-year period on a wide variety of different computers. The earliest ones were written in several different varieties of Basic as found on DEC PDP-8 computers. Some were written in Basic-Plus for the PDP-11. Others were written on large timesharing computers, Honeywell and Control Data, in particular. Still others were written on various microcomputers such as the Altair and TRS-80.

Two students, David and James Grosjean, converted all these programs to the Timex Sinclair 1000. Below are some of their notes about the problems and approaches in converting these programs. As you will see from these notes, there is a great deal to be learned by converting programs from one computer to another. After absorbing this material, you should be able to pick up a copy of a general personal computing magazine such as *Creative Computing* or a book written for another computer and convert many of the programs to run on your Timex Sinclair 1000.

Notes by David and James Grosjean

In translating the programs in this book to the Timex Sinclair 1000, we encountered several problems. The most common was that the 1000 lacked some of the functions that were used in the original programs which were written on other computers.

First, the 1000 does not have the commands READ, DATA, and RESTORE. Data that cannot be generated by a function are generally stored in a DATA statement. For example,

```
100 DATA 3, 6, 7, 13, 14, 17, 18, 24
```

The computer accesses these numbers with the READ command. A line such as 110 READ X would assign X the value of the first number in the first DATA statement. The computer keeps track of which piece of data it is processing, so each time it comes to a READ command, it looks at the next one. A program might look like this.

```
10 READ X
20 PRINT X*2
30 GOTO 10
30 DATA 3, 6, 7, 13, 14, 17, 18, 24
```

When run, this program would produce the following results: 6, 12, 14, 26, 34, 36, and 48. After it has printed these results, the computer will run out of data and stop.

DATA statements can also contain strings. For instance,

```
10 READ X$
20 PRINT "HI, "X$
30 GOTO 10
40 DATA "JOHN","TIMOTHY", "KAREN"
```

This program produces as output

```
HI, JOHN
HI, TIMOTHY
HI, KAREN
```

and then stops because it has run out of data. Some computers do not need quotation marks around the letters.

DATA statements can be put anywhere in the program and a program can have as many data statements as needed. The computer reads the data from the first statement and continues on to read additional ones, if there are any. When the computer runs out of data, the RESTORE command can be used to repeat the reading from the beginning.

```
10 FOR A=1 TO 8
20 READ X
30 PRINT X*2
40 NEXT I
50 RESTORE
60 GOTO 10
70 DATA 3, 6, 7, 13, 14, 17, 18, 24
```

This program prints 6, 12, 14, 26, 28, 34, 36, 48, 6, 12, 14, 26, 28, 34, 36, 48, and so on, until you stop it by hitting the BREAK key.

The 1000 does not have these commands, but there are several ways to simulate them. The program "Sets" illustrates the way we chose to use throughout the book.

The second "Sets" program compares a regular set of numbers (generated in a FOR loop) with a set that has no obvious pattern. The original program looks like this:

```
 7 PRINT "THE INTERSECTION OF SETS X AND Y IS:"
10 READ X
20 FOR Y=2 TO 29 STEP 3
30 IF X=Y THEN 70
40 NEXT Y
50 GO TO 10
70 PRINT X
80 GO TO 10
90 DATA 2,3,8,9,14,15,20,21,26,27
99 END
```

The original program reads the numbers of one set out of the DATA statement which is placed at the end of the program, and compares it to the set produced by the loop. This was simulated by putting the numbers separated by commas and ending with a comma, in a REM statement at the beginning of the program. The PEEK function is used to "READ" the number out of the REM statement and puts it into a string. Then the VALue of that string is assigned to X which is compared to the loop.

```
    1 REM 2,3,8,9,14,15,20,21,26,
27,
    2 LET A$=""
    4 PRINT "PROGRAM SETS-1"
    5 PRINT
    6 PRINT "THE INTERSECTION OF
SETS X AND Y IS:"
    8 LET I=16514
   10 LET A$=""
   12 LET A$=A$+CHR$ PEEK I
   16 LET I=I+1
   18 IF PEEK I=118 THEN STOP
   20 IF PEEK I<>26 THEN GOTO 12
   25 LET I=I+1
   30 LET X=VAL A$
   40 FOR Y=2 TO 29 STEP 3
   50 IF X=Y THEN GOTO 80
   60 NEXT Y
   70 GOTO 10
   80 PRINT X
   90 GOTO 10
```

Here is a line-by-line explanation:

1: This is our simulated DATA statement. We put it at the beginning of the program so that we will always know what the starting address is. This is the simplest place to put it because nothing precedes this line; therefore, the address is the same in all programs. The address right after the REM, the address of the first number is 16514.

4-6: Prints the display.

8: This is the address of the first character after the REM in line 1.

10: This line has two functions. It defines A$ the first time through and then later clears A$ of any previous numbers assigned to it during the read routine.

12: This line adds the next character in the REM statement to A$ so multi-digit numbers can be used.

16: This line increments the address, so that the computer will look at the next byte instead of the same one.

20: This checks to see if the next character in the REM statement is a comma. If it is not, the computer will goto to line 12 and continue to add numbers to A$.

25: This increments the address so that the computer will skip over the comma.

30: This assigns to X the VALue of A$. Lines 20 and 25, assure that there are no commas in A$ to confuse the computer. (Commas cannot be interpreted into anything mathematical.) A further explanation and use of VAL is presented in "Roots."

40-60: This is the loop of Y, which compares the value of X to each value in set Y. If the value of X matches any of the members of set Y, X will be printed.

65: This checks to see if the end of the REM statement has been reached. If it has, the program will stop.

70: This sends the computer back to line 10 where the program will get the next value of X.

When a set cannot be simulated with a loop, two REM (DATA) statements are needed. In order to use the second REM statement, we must calculate the address of the first character of the second REM line (in this case 16546 as indicated in line 40). We begin with address 16514, which is the address of the first character (2) after the REM in line 1. Then starting with the second character, we add 1 to 16514 for each subsequent character in the first REM line. There are 25 characters to be added. Next, we add 1 for the ENTER character at the end of the program line (you cannot see it). and then add another 2 for the line number of the next statement and another 2 for the two bytes between the line number and content of the line which tell the computer the length of the line. Then we add 1 more to get the address of the next character. In other words, take the first character of the first REM statement, add to this the number of characters in the REM line after the first character, add 1 for the ENTER character, 2 for the next line number, 2 for the line length bytes, 1 for the second REM, and 1 more to obtain the address of the first character after the second REM.

To use the second REM (DATA) statement, we simply duplicate the routine used for reading 1 REM and use it with 2 REM. See lines 40-120.

```
  1 REM 2,3,8,9,14,15,20,21,26,
27,
  2 REM 2,5,8,11,14,17,20,23,26
,29,
  4 PRINT "PROGRAM SETS-1"
  5 PRINT
  6 PRINT "THE INTERSECTION OF
SETS X AND Y IS:"
  8 LET I=16514
 10 LET A$=""
 12 LET A$=A$+CHR$ PEEK I
 16 LET I=I+1
 20 IF PEEK I<>26 THEN GOTO 12
 25 LET I=I+1
 30 LET X=VAL A$
 40 LET H=16546
 45 FOR S=1 TO 10
 50 LET B$=""
 60 LET B$=B$+CHR$ PEEK H
 70 LET H=H+1
 80 IF PEEK H<>26 THEN GOTO 60
 90 LET H=H+1
110 IF X=VAL B$ THEN PRINT X
120 NEXT S
130 GOTO 10
```

Another command not found on the Timex 1000 is DEF FNA (X). This DEFines a FunctioN named A (functions can be named by any letter). A sample program would read:

```
10 DEF FNA(X)=X*2
    .
    .
    .
    .
100 LET C=FNA(4)
```

C would equal B because the number in parentheses (4) in line 100 is substituted into the equation as X so C=4*2 or 8.

By using the VAL function of the 1000, DEF FNA can be simulated. The first line of the original version of "Roots" read 100 DEF FNA(X)=X-4. We replaced this statement with 100 LET A$="X-4". Then, whenever a line such as

170 LET Z=FNA(6)

came up, we just replaced it with two lines. The first sets X (or whatever variable was used in the formula) equal to the number or variable in parentheses. The second assigns the variable to VAL A$ instead of FNA(6). The conversion for the sample line above would look like this:

170 LET X=6
175 LET Z=VAL A$

The following line presented a special problem:

```
140 IF SGN(FNA(I))=SGN(FNA(I+I)) THEN GOTO 300
```

The two FNA expressions had to be evaluated separately before they could be used in the line. We replaced this line with lines 125 through 138. Line 125 LETs X=I and line 130 LETs A=VAL A$, or X-4. X=I, so A equals I-4. Lines 135 and 136 do basically the same thing except that the value of X is changed.

Anything that can be evaluated mathematically can be put into A$ and then read using VAL. For example, any letters in A$ must be variables that have already been defined. Mathematical functions such as SIN, COS, and ABS can be used. For a further explanation of strings and VAL see the chapter on strings in your manual.

Third, the Timex 1000 cannot use multi-statement lines. For example, a line might read

```
100 IF X=15 THEN LET V=5:LET A=B:GOTO 150
```

The simple solution to this would be to divide the line into several lines, but multi-statement lines starting with IF are tricky. Normally, the computer evaluates the IF statement, and, if it is true, executes the rest of the line. If the statement is false, the computer skips the rest of the line and drop to the next. So when a line is divided, this must be taken into account. Line 100 above, could be adapted as follows:

```
100 IF X=15 THEN GOTO 104
102 GOTO 110
104 LET V=5
106 LET A=B
108 GOTO 150
110 . . . rest of program
```

or

```
100 IF X<>15 THEN GOTO 108
102 LET V=5
104 LET A=B
106 GOTO 150
108 . . . rest of program
```

A few programs, like "Expon," use LOG. Simply replace LOG with LN. For example, you might find a line like this:

```
10 PRINT LOG(X)
```

On the 1000 you would change it to:

```
10 PRINT LN X
```

Some Basic use lines such as

10 A=0

On the 1000 this line must start with LET:

10 LET A=0.

Some Basics use symbols which are not found on the Timex 1000. For example, an up arrow (↑) or three dots in triangular form mean "raise to the power of." The equivalent on the 1000 is **. So check the function of the symbol in the original program and substitute the 1000 symbol.

The RND function works differently on just about every computer. Expressions such as RND(0) and RND(1) both translate to just RND on the Timex 1000. Expressions such as INT(RND(0)*X+1) or INT(RND(1)*X+1. (X is an integer or variable.) The TRS80 also has RND(X) where X is an integer. This takes a random integer from 1 to X. This also should be translated to INT(RND*X)+1.

Occasionally a program will have a line like

100 LET A=B=C=D=0

which would make A, B, C, and D equal to 0, but this line will not work on the 1000. The computer will first compare A to B. Since it does not know what B (or C or D) is, the computer will give an error code of 2 which means that it does not know what the value of the variable is. This line should be split up into a series of LET statements:

10 LET A=0
20 LET B=0
30 LET C=0
40 LET D=0

Some programs contain lines such as

```
10 ON X GOTO 100, 200, 300
```

In these lines, if X=1, the computer goes to line 100. If X=2, the computer goes to the second line number, 200, and so on. These can be translated to

```
10 GOTO X*100
```

or

```
10 GOSUB X*100
```

Sometimes this cannot be used with a line, as in

```
10 ON X GOTO 50, 135, 255
```

It can be translated to

```
10 IF X=1 THEN GOTO 50
20 IF X=2 THEN GOTO 135
30 IF X=3 THEN GOTO 255
```

or

```
10 GOTO ((X=1)*50+(X=2)*135+(X=3)*255)
```

This second method uses Boolean operations. First, the computer evaluates X. If X=1, then (X=1) would yield 1, which, to the computer, means true, and multiplies by 50. The other expressions would yield 0, which means false, and thus the total would be 50 and the computer would GOTO 50.

Finally, you will find programs written for other computers which use CALL, USR, POKE, or PEEK. These cannot be translated easily even though the Timex 1000 has USR, POKE, and PEEK. You must first determine how these functions work on the original computer and then figure out a way to make the 1000 do the same thing. At times a totally different approach may be required, but that's part of the challenge (and the fun) of programming.

1

Drill and Practice

Throughout life, there are certain things that simply must be memorized. Obvious things that fall into this category are the addition and multiplication tables, the spelling and meaning of words, how to tell time, and the monetary system.

However, depending upon one's chosen profession, there are many other things to be memorized. A doctor must know what diseases match what symptoms. A chemist must know the gas laws, the properties of elements and so on. A pilot must instantaneously know the meaning of readings on scores of instruments.

To memorize a set of facts, you must go over them again and again and keep trying different variations. Here is where the computer comes in. It is able to present randomly scores of different problems to you for as long as you wish. Some programs will automatically adjust to your level of competence and will grade you; other programs simply present the problems and leave the grading up to you.

There are four programs in this chapter and two in the Science chapter which present material in a drill and practice format. Examine the methods used in these programs and then make up some drill programs of your own for subjects with which you are having trouble, or make up programs for other members of your family.

Addition Practice

This program demonstrates a simplified addition drill and practice routine. This type of drill is sometimes called computer assisted instruction (CAI), although CAI can also apply to tutorial and other approaches as well.

When the program is run, it will first ask "How many digits?" You enter a number and each addend will contain that many or fewer digits.

The program is set up to provide ten practice problems. The program presents each problem in turn. The program will not proceed to the next problem until the current one has been answered correctly. After the last problem is answered correctly, the score is printed with an appropriate comment.

There are many improvements and extensions possible in a program like this one. For example, you could change it to ask the user the number of problems he wishes to do. This would require a change in Statement 25 which currently causes ten problems to be presented. You would also have to change Statements 160 and 165 which present the score.

You might also want to modify the program so it tells the user the correct answer after a problem has been answered incorrectly two (or three) times.

A more complicated modification would be to change the program to present different kinds of arithmetic problems such as subtraction, multiplication, and division.

Some of these modifications have been made in the next program.

```
   5 REM - CAI ADDITION, 10 PROB
LEMS
  10 PRINT "HOW MANY DIGITS?"
  20 INPUT N
  22 LET U=0
  25 FOR X=1 TO 10
  28 LET Q=0
  30 LET A=INT (10**N*RND)
  40 LET B=INT (10**N*RND)
  42 LET A$=STR$ A
  44 LET B$=STR$ B
  46 IF LEN A$=N-1 THEN LET A$=A
$+"0"
  48 IF LEN B$=N-1 THEN LET B$=B
$+"0"
  50 LET A=VAL A$
  52 LET B=VAL B$
  55 PRINT
  56 IF PEEK 16442<6 THEN PAUSE
300
  58 IF PEEK 16442<6 THEN CLS
```

```
  60 PRINT "    ";A
  70 PRINT "   +";B
  80 PRINT "   ---------"( TO N+3)
  90 PRINT "   ";
 100 INPUT G
 105 PRINT G
 110 IF G=A+B THEN GOTO 140
 115 GOSUB 200
 120 PRINT "WHAT?? TRY AGAIN"
 130 GOTO 55
 140 PRINT "THAT'S RIGHT. TRY AN
OTHER."
 150 NEXT X
 155 PRINT
 160 PRINT "YOU GOT ";10-W;" COR
RECT THE FIRST","TIME"
 165 IF W<=3 THEN GOTO 180
 170 PRINT "BUT YOU MISSED ";W
 175 STOP
 180 PRINT "GOOD WORK."
 185 STOP
 200 LET Q=Q+1
 205 IF Q>1 THEN GOTO 120
 210 LET W=W+1
 215 RETURN

HOW MANY DIGITS?

    1045
   +8459
   -----
   9504
THAT'S RIGHT. TRY ANOTHER.

    4462
   +4665
   -----
   9127
THAT'S RIGHT. TRY ANOTHER.

    9943
   +5765
   -----
   15608
WHAT?? TRY AGAIN


    9943
   +5765
   -----
   15708
THAT'S RIGHT. TRY ANOTHER.

    2431
   +2347
   -----
   4778
THAT'S RIGHT. TRY ANOTHER.
```

Addition Practice, Adjusted by Grade Level

One of the major disadvantages with many drill and practice exercises is that they tend to be either boring or frustrating, depending upon the ability of the user relative to the level of the material. To compensate for this, a method is needed which will adjust the difficulty of the problems to the ability of the user.

Ideally, such a system would weigh the most recent performance most heavily but would not ignore previous performance. It should allow a user to advance to more difficult problems than his current mastery level. It should also continue to give some practice on problems already mastered.

Some commercial software packages approach these goals along traditional lines, i.e., determine in which type of problems a student should receive practice by using a complicated computer managed instruction score recording and adjustment system.

The approach here is more innovative; it uses a single measure for each type of problem—call it "estimated grade level"—which meets all of the objectives stated above.

How does it work? The most recent problem presented counts 10% of the overall score if it was answered correctly and was over the current user grade level, or if it was answered incorrectly and was under the current user grade level. Otherwise it is ignored. This may be easier to visualize in the form of a chart:

		Answer	
		Right	*Wrong*
Problem	*Higher than grade level*	Raise student grade level	Ignore
	Lower than grade level	Ignore	Lower student grade level

At first glance this might look complex and somewhat goofy, however, what it really means is that a student is rewarded for doing a problem beyond his grade level but he is not penalized if he cannot do it. On the other hand he is penalized if he cannot do a problem lower than his grade level, but is not rewarded for doing one lower.

Each problem affects the estimated grade level a little bit, with the most recent problems being weighed the most heavily. If the current grade level of a student is L and the level of the most recent problem to be averaged in is P,

then the averaging formula is simply:

$$L = .9L + .1P$$

The remaining task before a program can be written is to assign a grade level to each problem presented. Unfortunately, this will vary depending upon the local school system, the textbook used, and the teaching method. Also a huge data base can not be stored in a small computer, so it is desirable to devise a simple method of determining grade level for different problems. One straightforward approach is to present problems up to one-half a grade level over and under where the student currently is. Thus the overall range of problems for a student at grade level 3.2 would be 2.7 to 3.7.

How do we generate the right problems? Consider one type of skill, vertical addition. It is normally introduced in the first grade and continues through Grade 4 (actually 4.9). The simplest problem in this program is 1 + 1 and the most difficult is 999 + 999. Since learning is not a linear process (it is slow at first, and then progresses rapidly), an exponential formula can be used. For example:

$$\text{Addend} = 1.73 \times (\text{Grade level})^4$$

or

$$\text{Grade level} = \sqrt[4]{\text{Addend}/1.73}$$

This means that students at various grade levels will be working with the following maximum addends:

Grade level	*Addend*
1.0	1
2.0	27
3.0	140
4.0	442
4.9	997

Now it is a relatively straightforward, although somewhat tedious, matter to tie all these elements together in a computer program.

A few notes about the program. The variable G2 is the problem grade level that is always within one-half of a grade level of the current student level, G1. The complicated mess in Statement 350 produces a moving grade level average.

The recording of the grade level and carrying it over to the next lesson is a manual process. On a computer system with a permanent mass storage device, this would be kept on the system.

There are many possible changes and extensions to this program. For example it could present different types of problems such as horizontal addition, vertical and horizontal subtraction, as well as multiplication, division and fraction problems.

```
    5 LET W=0
   10 RAND
   15 PRINT "HI. TO STOP, INPUT 9
999 AS YOUR ANSWER."
   20 PRINT "WHAT IS YOUR GRADE L
EVEL? ";
   30 INPUT G1
   35 PRINT G1
   40 LET G2=G1-.5*RND
   50 LET A=INT (2*1.73*G2**4)
   60 LET A=INT (100*G2*RND)
   70 IF A>A THEN GOTO 60
   80 LET B=A-A
   90 PAUSE 250
   95 CLS
  110 PRINT TAB 8-LEN STR$ B;B
  120 PRINT " + ";TAB 8-LEN STR$
A;A
  140 LET A=A+B
  200 PRINT "----------"
  210 INPUT G
  215 PRINT TAB 8-LEN STR$ G;G
  220 IF G=A THEN GOTO 310
  225 IF G=9999 THEN GOTO 500
  230 LET W=W+1
  240 IF W>1 THEN GOTO 270
  250 PRINT "WRONG. TRY AGAIN."
  260 GOTO 100
  270 PRINT "YOU MISSED THAT ONE
TWICE."
  280 PRINT "THE CORRECT ANSWER I
S ";A
  285 LET W=0
  290 IF G2>G1 THEN GOTO 400
  300 GOTO 330
  310 LET W=0
  320 PRINT "CORRECT."
  330 IF G2<G1 THEN GOTO 400
  350 LET G1=.9*G1+.1*SQR (SQR (A
/2/1.73))
  400 PRINT
  405 PRINT "HERE'S ANOTHER..."
  410 GOTO 40
  500 PRINT "OKAY. SO LONG. HOPE
YOU HAD FUN."
  510 PRINT "YOUR GRADE LEVEL IS
NOW ";G1
```

```
HI. TO STOP, INPUT 9999 AS YOUR
ANSWER.
WHAT IS YOUR GRADE LEVEL? 8


   13542
 +    24
---------
   13566
CORRECT.

HERE'S ANOTHER...


   12300
 +   356
---------
   12656
CORRECT.

HERE'S ANOTHER...


   12767
 +   415
---------
   13181
WRONG. TRY AGAIN.
   12767
 +   415
---------
   13183
YOU MISSED THAT ONE TWICE.
THE CORRECT ANSWER IS 13182

HERE'S ANOTHER...
```

Time/Speed/Distance Problems

As well as being able to present simple numerical drill and practice problems, the computer can present word problems for solution as well.

In this program the formula relating time, speed and distance was applied to a problem involving both a car and train.

The problem can be stated as follows. A car traveling C miles per hour (computer generates an integer 40 through 65) can make a certain trip in D hours (computer generates an integer 5 through 20) less than a train traveling at T mph (computer generates an integer 20 through 39). How long does the trip take by car? When the two simultaneous equations are solved they produce the single equation for the answer shown in Line 60.

Notice the calculation in Line 65. This calculates the percent difference between the actual answer and the one entered by the user.

Notice also that the computer calculates the correct answer in Line 60 and prints it (on the screen) in Line 90. This answer may have many decimal places; as the program is written it is not rounded off at all. A good change that you can put in the program is to have it round off the answer to one or two decimal places.

More important than changing this program is to consider other problems that can be used as the basis for this kind of drill and practice exercise. Teachers, for example, might wish to have students write drill and practice programs on their own. Different problems could be given to one or a small group of students to serve as the basis for a program.

After the programs are written, students can try out the programs of other class members. This approach ensures that each student not only understands the type of problem assigned to him, but also gets practice in solving other problem types as well. This is an effective technique for stimulating interest as well as for learning how to solve word problems.

```
  5 RAND
 10 LET C=INT (RND*25)+40
 15 LET D=INT (RND*15)+5
 20 LET T=INT (RND*19)+20
 25 PRINT "A CAR TRAVELING ";C;
" MPH CAN"
 30 PRINT "MAKE A CERTAIN TRIP
IN ";D;" HOURS"
 35 PRINT "LESS THAN A TRAIN TR
AVELING AT"
 40 PRINT T;" MPH.  HOW LONG DO
ES THE TRIP"
 45 PRINT "TAKE BY CAR?"
 50 INPUT A
 55 PRINT A
 60 LET V=D*T/(C-T)
 65 LET E=INT (ABS ((V-A)*100/A
)+.5)
 70 IF E>5 THEN GOTO 85
 75 PRINT "GOOD. ANSWER WITHIN
";E;" PERCENT."
 80 GOTO 90
 85 PRINT "SORRY. YOU WERE OFF
BY ";E,"PERCENT."
 90 PRINT "CORRECT ANSWER IS ";
V;" HOURS."
```

```
A CAR TRAVELING 44 MPH CAN
MAKE A CERTAIN TRIP IN 11 HOURS
LESS THAN A TRAIN TRAVELING AT
28 MPH.  HOW LONG DOES THE TRIP
TAKE BY CAR?
25
SORRY. YOU WERE OFF BY 23
PERCENT.
CORRECT ANSWER IS 19.25 HOURS.
```

```
A CAR TRAVELING 42 MPH CAN
MAKE A CERTAIN TRIP IN 15 HOURS
LESS THAN A TRAIN TRAVELING AT
33 MPH.  HOW LONG DOES THE TRIP
TAKE BY CAR?
55
GOOD. ANSWER WITHIN 0 PERCENT.
CORRECT ANSWER IS 55 HOURS.
```

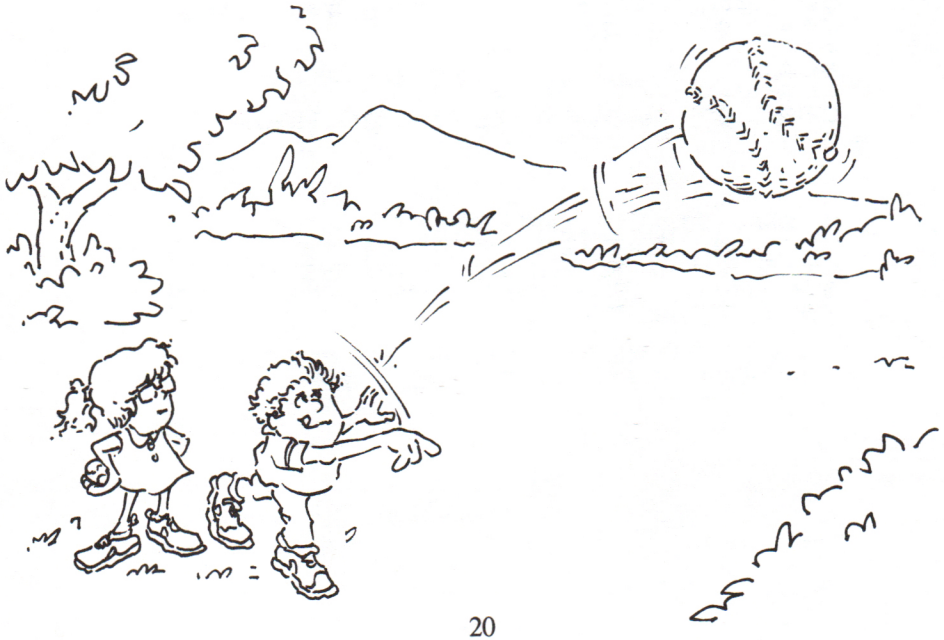
Kinematics Problems

Kinematics relates to the dynamics of motion of bodies apart from considerations of mass and force. Like many other types of problems, these can be generated and presented by the computer to provide practice in solving them.

This program presents a simple kinematics problem for solution. The computer generates a new value for each problem. The problem is as follows. A ball (or any other object) is thrown up at velocity V meters per second (computer generates an integer between 5 and 40). The user must then calculate three factors about the resulting flight of the ball: maximum height, time until it returns, and velocity after T seconds (computer generates a time less than the total flight time).

The key benefit in using a computer to present problems of this type is motivation. The calculations required of the user are no different than those in the back of a chapter in a book or those on homework assignments. However, answering them when they are presented by the computer seems to make it more of a challenge and, frankly, more fun.

The computer program checks each of your responses to see if it is within 15% of the correct answer. If it is, your answer is considered correct. You may wish to change this percentage to require more or less accurate calculations. You may also wish to change the computer calculations to round off to one or two decimal places.



```
 100 LET Q=0
 110 LET V=6+INT (35*RND)
 111 PRINT "A BALL IS THROWN UPW
ARDS AT ";V;"METERS PER SECOND"
 112 PRINT
 115 LET A=.05*V**2
 116 PRINT "HOW HIGH WILL IT GO
(IN METERS)?"
 117 GOSUB 500
 120 LET A=V/5
 122 PRINT "HOW LONG UNTIL IT RE
TURNS (IN   SECONDS)?  ";
 124 GOSUB 500
 130 LET T=2+INT (2*V*RND)/10
 132 LET A=V-10*T
 134 PRINT "WHAT WILL ITS VELOCI
TY BE AFTER ";T;" SECONDS?  ";
 136 GOSUB 500
 140 PRINT
 150 PRINT Q;" RIGHT OUT OF 3.";
 160 IF Q>1 THEN PRINT "  NOT BA
D."
 180 STOP
 500 INPUT G
 501 PRINT G
 502 IF ABS ((G-A)/A)<.15 THEN G
OTO 510
 504 PRINT "NOT EVEN CLOSE...."
 506 GOTO 512
 510 PRINT "CLOSE ENOUGH."
 511 LET Q=Q+1
 512 PRINT "CORRECT ANSWER IS ";
A
 520 PRINT
 530 RETURN
```

```
A BALL IS THROWN UPWARDS AT 13
METERS PER SECOND

HOW HIGH WILL IT GO (IN METERS)?
8
CLOSE ENOUGH.
CORRECT ANSWER IS 8.45

HOW LONG UNTIL IT RETURNS (IN
SECONDS)?  3
NOT EVEN CLOSE....
CORRECT ANSWER IS 2.6

WHAT WILL ITS VELOCITY BE AFTER
1.7 SECONDS?  -4
CLOSE ENOUGH.
CORRECT ANSWER IS -4

2 RIGHT OUT OF 3.  NOT BAD.
```

"This nano-computer is great, but working the keyboard is a real problem!"

2

Problem Solving

In many courses in school, the textbooks present a great variety of devices for solving the problems that have been neatly grouped together at the end of each chapter. Typically these devices consist of formulae, equations, rules and theorems. After a careful study of these devices, teachers give exams which test your ability to recall them.

But what do you do if you are faced with the more realistic situation of not being told what device is likely to solve which problem or, worse yet, of having forgotten how to use a technique altogether? Is all hope lost? Of course not, although some people seem to believe that it is.

In this chapter, several of these nasty devices mentioned above are presented. For example, there are devices written into computer programs that will solve a quadratic or exponential equation and others that will calculate the roots and draw a plot of any function.

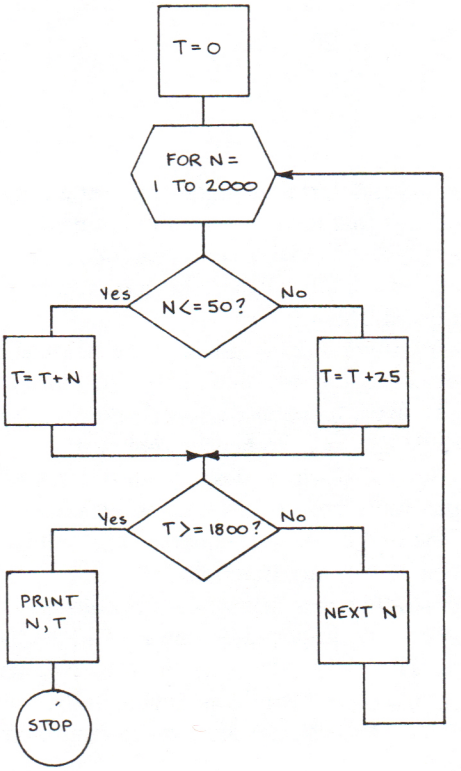
However, you must remember that while these devices are useful in solving certain problems, the really important thing is to understand the underlying logic and approach. Then when it comes time to solve real world problems you will be better prepared to face them. Incidentally, many of the methods and devices presented in this chapter are used in later chapters to solve other kinds of problems.

How Many Tickets?

Here is a problem. At a school raffle to raise money, the organizers have as a prize an electronic game for which they paid $18.00. To add interest to the raffle, the organizers have decided to sell tickets for an amount (in cents) equal to the number on the ticket for tickets numbered 1 to 50. For ticket numbers over 50, the price is 25 cents each. The organizers want to know how many tickets they must sell to exactly break even.

It is probably easiest to visualize a problem of this sort with a flowchart. In the flowchart, T will equal the total money collected and will increase as more tickets are sold. The ticket number is N. When T equals or exceeds $18.00, N will be the answer.

Note that the flowchart has two logical branching points (IF statements in the program). The first compares the current ticket number to 50; if it is less, the ticket number is added to the total whereas if it is greater than 50, the total is increased by 25 cents.



The second branch point compares the total amount collected, T, to $18.00 (actually 1800 cents). If T is equal to or greater than 1800, the break even point has been reached and the values of N and T are printed (on the screen).

A problem of this kind can be done by hand, however, because of the repetitive additions it is quite tedious. Also, doing it by hand frequently leads to an answer of 72 rather than the correct answer of 71. Try it yourself and see what you get.

```
  5 PRINT "PROGRAM TICKET"
  6 PRINT
 10 LET T=0
 20 FOR N=1 TO 2000
 30 IF N<=50 THEN GOTO 60
 40 LET T=T+25
 50 GOTO 70
 60 LET T=T+N
 70 IF T>=1800 THEN GOTO 90
 80 NEXT N
 90 PRINT N;" TICKETS SOLD TO C
OLLECT","EXACTLY $";T/100
```

```
PROGRAM TICKET

71 TICKETS SOLD TO COLLECT
EXACTLY $18
```

Many problems can be solved quickly and correctly with a computer using logical analysis and a flowchart. More complex problems may have to be broken down into additional steps and require a longer flowchart, but the approach is fundamentally the same.

Here are two problems for you to solve.

The diameter of a long-playing record is 12 inches. The unused center has a diameter of 4 inches and there is a smooth outer edge 1/2 inch wide around the recording. If there are 91 grooves to the inch, how far does the needle move during the actual playing of the recording?

A movie theater charges $2.50 for an adult admission and $1.00 for a child. At closing, the cashier counted 385 ticket stubs and had $626.50 in cash. How many children entered?

Drinking and High Blood Pressure

This program illustrates how several simple equations can be put in a computer program to solve a more difficult overall problem.

Here is the problem. In a survey of 1000 adults, it was found that 35 had high blood pressure. Of those with high blood pressure, 80% drink 15 oz. or more of alcohol per week. Of those without high blood pressure, 60% drink a similar amount. What percent of drinkers and non-drinkers have high blood pressure?

This problem requires the solution of several simple equations. They could all be combined into one large equation, but it may be easier to understand the approach (and change variables later on) by writing a program with five separate equations.

If H equals the number of people with high blood pressure, then H = 35 (Line 10). Letting H1 equal the number of people with high blood pressure who drink leads to H1 = .8 x H (Line 20).

Letting L1 equal the number of people with low blood pressure who drink yields L1 = .6 x (1000 - H). Then, the total number of drinkers, D = H1 + L1.

Finally, the percentage of drinkers with high blood pressure is X = H1 x 100 / D.

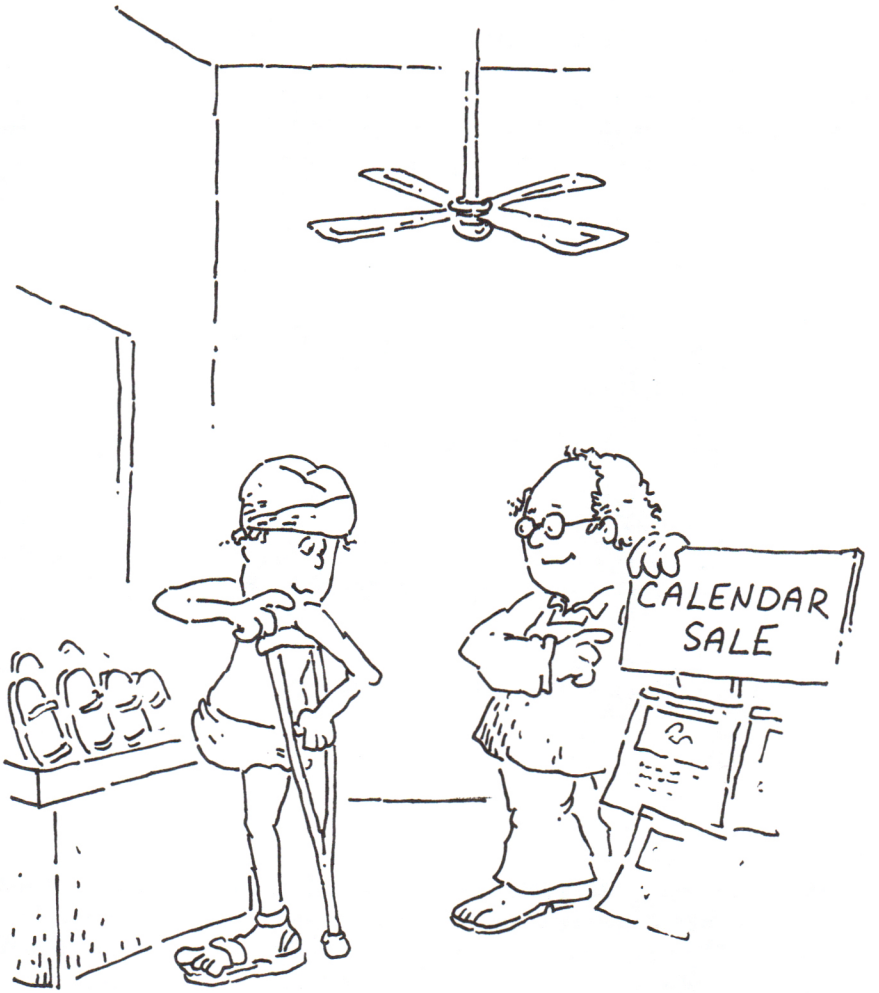
The program solves the problem in a jiffy. The solution for this type of problem can be easily written directly in Basic without any need for a flowchart or detailed analysis. Recognizing this type of problem readily will save a great deal of pencil pushing time.

```
  5 PRINT "PROGRAM DRINKR"
  6 PRINT
 10 LET H=35
 20 LET H1=.8*H
 30 LET L1=.6*(1000-H)
 40 LET D=H1+L1
 50 LET X=H*100/D
 60 PRINT X;" PERCENT DRINKERS
HAVE","HIGH BLOOD PRESSURE."
 70 PRINT H*100/1000;" PERCENT
OF ALL PEOPLE HAVE","HIGH BLOOD
PRESSURE."

SAMPLE RUN

PROGRAM DRINKR

5.7660626 PERCENT DRINKERS HAVE
HIGH BLOOD PRESSURE.
3.5 PERCENT OF ALL PEOPLE HAVE
HIGH BLOOD PRESSURE.
```

Here is a problem that doesn't require a single equation but makes use of the approaches discussed so far in this chapter. Can you solve it with three Basic statements?

In early January, a shopkeeper marked down some calendars from $2.00 to a lower price. He sold his entire stock in one day for $603.77. How many did he have?

Here's another easy one. A town in India has a population of 20,000 people. Five percent of them are one-legged and half of the others go barefoot. How many sandals are worn in the town?



Two Simultaneous Equations

So far in this chapter, only problems with linear equations have been considered. But the computer can be used to solve much more difficult equations. In fact, it is in problems involving second and third degree equations, exponentials, and the like where the computer really starts to pay off. Consider the following two simultaneous equations:

$$2^X = \frac{16y}{3} \qquad 3^X = 27y$$

It is not at all easy to solve these two equations by hand. But a simple Basic program can be written to solve the equations using trial and error. This is sometimes referred to as a brute force approach because every possible combination of numbers between an upper and lower limit is tried until a solution is reached or until the program runs out of values.

```
   5 PRINT "PROGRAM SIMUL"
   6 PRINT "SOLUTION TO TWO SIMU
LTANEOUS","EQUATIONS"
   7 PRINT
  10 FOR X=1 TO 100
  20 FOR Y=1 TO 100
  30 IF 2**X<>16*Y/3 AND 3**X<>2
7*Y THEN GOTO 60
  40 PRINT "X=";X,"Y=";Y
  50 STOP
  60 NEXT Y
  70 NEXT X
  80 PRINT "NO INTEGER SOLUTION"
```

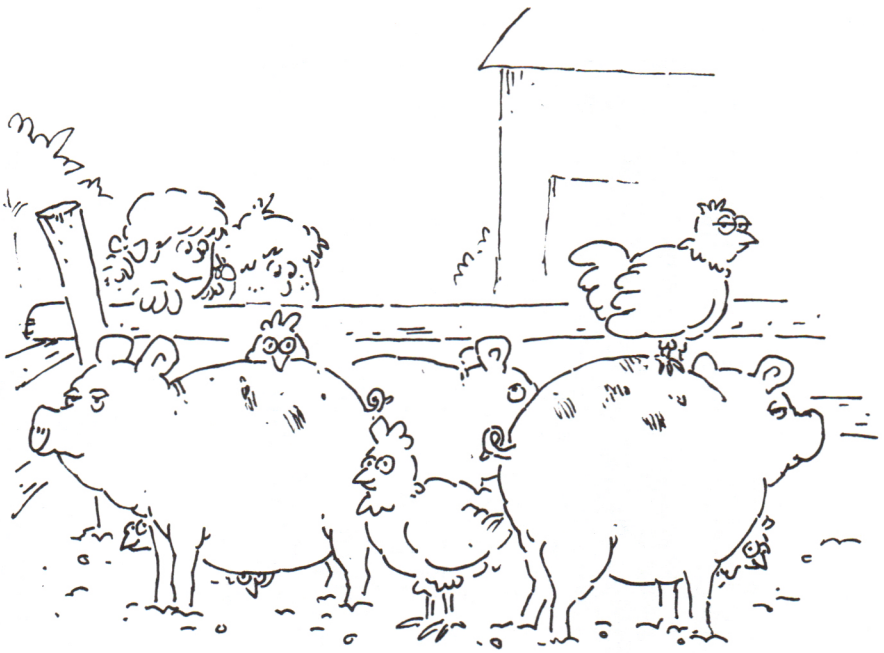
```
PROGRAM SIMUL
SOLUTION TO TWO SIMULTANEOUS
EQUATIONS

X=4                    Y=3
```

In this particular case, a solution is reached rather quickly with x = 4 and y = 3. However, if in the second equation the y coefficient is changed slightly from 27 to 28, the computer will try 10,000 possible solutions before finally concluding that no integer solution exists—at least within the range of 0 to 100.

```
10 FOR X=1 TO 10
20 FOR Y=1 TO 10
30 IF 3*X<>28*Y THEN GOTO 60

PROGRAM SIMUL
SOLUTION TO TWO SIMULTANEOUS
EQUATIONS

NO INTEGER SOLUTION
```

Although the trial and error (brute force) approach is widely used, it is highly inefficient. In general, a systematic or guided trial and error approach is preferable to one that simply tries every possible solution. However, for some problems the simple "try every value" approach may be appropriate. (A comprehensive discussion of trial and error approaches can be found on pp 36-40 of *Computers in Mathematics: A Sourcebook of Ideas.)*

Three problem solving approaches have been discussed so far. Remembering them, how would you do this problem? A boy and his sister visited a farm where they saw a pen filled with pigs and chickens. When they returned home, the boy observed that there were 18 animals in all, and his sister reported that she had counted a total of 50 legs. How many pigs were there in the pen?



Quadratic Equation Solver

For any values A, B, and C of a first degree quadratic equation ($Ax^2 + Bx + C = 0$), this program will compute the roots of the equation. The solution is based on the quadratic theorem which solves for roots with the following formula:

$$X = \frac{-B \pm \sqrt{B^2-4AC}}{2A}$$

Assuming that A, B, and C are real numbers, the following principles apply:

1. If B^2 - 4AC is positive, then the roots are real and unequal.
2. If B^2 - 4AC equals 0, then the roots are real and equal.
3. If B^2 - 4AC is negative, then the roots are imaginary and unequal.

The program takes into account all these possibilities and correctly identifies the type of roots along with their values for any set of coefficients.

Is this program useful by itself? Except for solving quadratic equations for algebra class, probably not. However, as a routine in a larger program to solve quadratic equations that might be encountered, it could be very useful.

```
  10 PRINT "QUADRATIC EQUATION S
OLVER"
  20 PRINT "PLEASE INPUT THE FOL
LOWING:"
  25 PRINT "VALUE OF A ";
  30 INPUT A
  35 PRINT A
  40 PRINT "VALUE OF B ";
  45 INPUT B
  50 PRINT B
  55 PRINT "VALUE OF C ";
  60 INPUT C
  65 PRINT C
  70 PRINT
  75 LET R=B**2-4*A*C
  80 IF A<>0 THEN GOTO 100
  85 PRINT "THIS IS A FIRST DEGR
EE EQUATION"
  90 STOP
 100 PRINT "THE ROOTS ARE ";
 105 IF R<0 THEN GOTO 145
 110 PRINT
 115 IF R=0 THEN GOTO 135
 120 PRINT (-B+SQR (R))/2*A
 125 PRINT (-B-SQR (R))/2*A
 130 STOP
 135 PRINT -B/2*A
 140 STOP
```

```
 145 PRINT "IMAGINARY"
 150 PRINT (-B/2*A);" + ";SQR (-
R)/2*A;" *I"
 155 PRINT (-B/2*A);" - ";SQR (-
R)/2*A;" *I"
```

```
QUADRATIC EQUATION SOLVER
PLEASE INPUT THE FOLLOWING:
VALUE OF A 1
VALUE OF B 2
VALUE OF C 1

THE ROOTS ARE
-1
```

```
QUADRATIC EQUATION SOLVER
PLEASE INPUT THE FOLLOWING:
VALUE OF A 0
VALUE OF B 2
VALUE OF C 4

THIS IS A FIRST DEGREE EQUATION
```

```
QUADRATIC EQUATION SOLVER
PLEASE INPUT THE FOLLOWING:
VALUE OF A 2
VALUE OF B 5
VALUE OF C 2

THE ROOTS ARE
-2
-8
```

```
QUADRATIC EQUATION SOLVER
PLEASE INPUT THE FOLLOWING:
VALUE OF A 4
VALUE OF B 2
VALUE OF C 4

THE ROOTS ARE IMAGINARY
-4 + 15.491933 *I
-4 - 15.491933 *I
```

Exponential Equation Solver

Another general routine for solving a particular type of equation is this one to solve for an exponent in an exponential equation.

Given the values A, B, m, and n, this program will solve for x in any exponential equation of the form:

$$A^{mx+n} = B$$

For example, the program will solve any of the following problems:

1. $5^X = 40$
2. $5^{3X+1} = 7.6$
3. $17^{X-3} = 8.12$
4. $11^{1-2X} = 247$

If you were to solve an exponential equation by hand, you would probably go through the following steps:

$5^X = 40$

$\log 5^X = \log 40$

$x \log 5 = \log 40$

$x = \log 40/\log 5$

$x = 1.6021/.6990 = 2.292$

However, in more generalized form, the solution for x is:

$$X = \frac{\frac{\log B}{\log A}}{M-(N/M)}$$

Note that the program to solve this problem is actually divided into two sub-programs. The first is a data loader program. It requires that data be entered in the following order: A, B, m, and n for each equation to be solved. If a coefficient is not present, it must be entered as a zero.

The data for the four problems listed above were entered into the loader program, then the main program was run to yield the four solutions.

The program as it is presented here can be improved in several ways. First, it always solves for four equations. How can you generalize it to solve for any number of equations? Second, if you were to make use of this routine in another program, you would not want a separate data loader; how could you get rid of it?.

EXPON-LOADER

```
 100 DIM A(16)
 200 FOR N=1 TO 16
 300 INPUT A(N)
 400 NEXT N
```

INPUT THE FOLLOWING NUMBERS:

```
5,40,1,0,5,7.6,3,1,17,8.12,1,-3,
11,247,-2,1
```

EXPON-MAIN PROGRAM

```
   1 FOR N=1 TO 4
 100 LET M=A(3+(N-1)*4)
 200 LET X=(LN A(2+(N-1)*4)/LN A
(1+(N-1)*4))/M-(A(4+(N-1)*4)/M)
 300 PRINT X
 400 NEXT N
```

```
2.2920297
.0867197B6
3.7392067
-0.6487967
```

Instructions:

1. Type in Loader program and Run it.
2. Enter the list of numbers.
3. Type in the Main program over the loader. Do not clear or run.
4. Type GOTO 1 to execute the program.

Roots of Any Function

This program will find the roots of a function, any function! The function may be linear, quadratic, cubic, trigonometric or any combination as long at it can be represented in the Basic language. The program as it appears here finds the roots between -20 and 20 although you can change these boundaries in Statement 120.

The method used involves evaluating the function at small incremental intervals, finding places where the value of the function changes sign and then, by successive approximations, finding the zero point. This approach borrows from Newton's method in the final narrowing down but, unlike Newton's method, will not fail to converge in the event one makes an unlucky first guess.

Before running the program you must first type in your function and set it equal to A$ in Statement 100. For example,

```
100 A$= "2*(X*X*X)+11*(X*X)-31*X-180"
100 A$= "X-4"
100 A$= "SIN X-.5"
```

Be sure to put your function inside quotation marks as shown in the examples above. Also, you may have to refer to the Basic manual with your system to see exactly how a function should be stated.

The routine used in this program is very powerful and could possibly be used as a subroutine in many other programs. However, bear in mind the functions are stated as string variables which is not normally the way they would be found in a larger program.

How can you use this program to help you solve this problem for x?

$$x = \sqrt{12 + \sqrt{12 + \sqrt{12 + \sqrt{12 + \sqrt{12 + \ldots}}}}}$$

```
  100 LET A$="2*(X*X*X)+11*(X*X)-
31*X-180"
  110 LET Z1=-200
  120 FOR I=-19.9 TO 20
  125 LET X=I
  130 LET A=VAL A$
  135 LET X=I+1
  136 LET B=VAL A$
  138 IF SGN A=SGN B THEN GOTO 30
0
  140 LET K=I
  150 LET J=I+1
  152 LET X=K
  154 LET A=VAL A$
```

```
 156 LET X=J
 158 LET B=VAL A$
 160 IF A<B THEN GOTO 200
 170 LET Z=K
 180 LET K=J
 190 LET J=Z
 200 LET Z=(K+J)/2
 205 LET X=Z
 210 IF VAL A$<0 THEN GOTO 240
 220 LET J=Z
 230 GOTO 250
 240 LET K=Z
 250 LET X=Z
 255 IF ABS (VAL A$)>5E-5 THEN G
OTO 200
 260 LET Z=SGN Z*INT (ABS Z*1000
0+.5)/10000
 270 IF Z=Z1 THEN GOTO 300
 280 PRINT "F(";Z;")=0"
 290 LET Z1=Z
 300 NEXT I
 310 STOP


F(-5)=0
F(-4.5)=0
F(4)=0




 100 LET A$="X-4"

F(4)=0




 100 LET A$="SIN X-.5"

F(-18.3259)=0
F(-16.2315)=0
F(-12.0428)=0
F(-9.9485)=0
F(-5.7596)=0
F(-3.6651)=0
F(0.5236)=0
F(2.6179)=0
F(6.8068)=0
F(8.9011)=0
F(13.09)=0
F(15.1844)=0
F(19.3732)=0
```

Plot Any Function

Here is a nifty program that will produce a plot of any function on the screen or printer.

When the program is run, it will ask you to type a new Statement 100 (which skips the initial instructions after you have seen them) and a Statement 200 which contains your function. Like the previous program, which finds the roots of any function, this one will plot any function. You must tell the program between what values you want the function plotted, i.e., a minimum and maximum value of the x coordinate. You also input the x plotting increment you wish.

As it appears here, the program does not allow the user to select the y coordinates; the program plots y values between -30 and 30.

Here are several functions you might want to try plotting:

Function	*X Limits*		*Increment*
`200 A$="2*X"`	-15	15	1
`200 A$="30*SIN X"`	-5	5	.25
`200 A$="X-X*X"`	-5	6	.5
`200 A$="30*EXP(-X*X/100)"`	-30	30	1.5
`200 A$="X*X-X"`	-5	6	1

The last function listed is the one plotted in the sample run with the program. Exponential functions are a great deal of fun and sometimes lead to unexpected and interesting results, particularly when combined with trigonometric functions. Experiment! Have fun!

```
 110 PRINT "TO PLOT Y=F(X), THE
USER MUST  TYPE:"
 130 PRINT "  100 GOTO 200"
 140 PRINT "  200 LET A$="""(THE E
XPRESSION"
 150 PRINT TAB 14;"FOR F(X))"""
 160 PRINT "RUN"
 170 STOP
 200 LET A$="X*X-X"
 210 PRINT
 220 PRINT "WHAT ARE THE INITIAL
 AND FINAL  VALUES OF X, AND THE
 INCREMENT  SIZE?  ";
 230 INPUT X1
 240 INPUT X2
 250 INPUT S
 260 PRINT X1;",";X2;",";S
 270 PAUSE 200
```

```
 280 CLS
 290 PRINT TAB 16;"X=";X1
 300 FOR X=X1 TO X2 STEP S
 310 IF ABS X<.000001 THEN GOTO
370
 330 LET Y=VAL A$+35
 340 IF Y>35 THEN GOTO 410
 350 PRINT TAB Y/2-3;"*";TAB 16;
"."
 360 GOTO 420
 380 PRINT "-Y. . . . . . . . .
. . . . . +Y";TAB (VAL A$+35)/2-
3;"*"
 390 PRINT "-30  -20  -10  0  +1
0  +20  +30"
 400 GOTO 420
 410 PRINT TAB 16;".";TAB Y/2-3;
"*"
 420 NEXT X
 430 PRINT TAB 16;"X=";X2
```

```
TO PLOT Y=F(X), THE USER MUST
TYPE:
 100 GOTO 200
 200 LET A$="(THE EXPRESSION
              FOR F(X))"
RUN
```

```
WHAT ARE THE INITIAL AND FINAL
VALUES OF X, AND THE INCREMENT
SIZE?  -5,6,1
```

```
                    X=-5
                    .                *
                    .          *
                    .     *
                    . *
                    .
                    *
-Y. . . . . . . . . . . . . . . . +Y
                   *
-30  -20  -10  0  +10  +20  +30
                   *.
                    .
                    *
                    . *
                    .     *
                    .          *
                    .                *
                    X=6
```

3

Sets and Repetitive Trials

For solving relatively simple problems, the computer may not be any help at all. In fact, it may take more time to write a program to solve a problem than it would to solve it by hand or with a calculator. This chapter should help you recognize problems that are suitable for computer solution and those that are not.

In some problems, you may think that the computer will not be any help. However, one thing writing a computer program will always do is force you to reason out the approach to solving the problem logically and precisely. The computer can't solve problems unless it is told exactly how to proceed; hence you must understand a problem completely before you can program it for the computer.

Several of the sections in this chapter discuss sets of data. While the sets used in the examples have relatively few elements or values, you should bear in mind that real world problems often have thousands or millions of pieces of data and the only practical way to solve problems of this size is with a computer. For example, consider how you would most efficiently schedule the shipments on a railroad train leaving Boston with 4000 diverse cargoes bound for Phoenix and 780 points in between. Now consider that there are 200 freight trains per day leaving Boston. Now add to that the 10,000 other trains leaving other cities every day that must use the same network of track and you can see that dealing with real world sets of data is no easy task.

Group of Girls and Boys

In the previous chapter we said that there may be some problems for which brute force trial and error is appropriate. This would be the case if the problems were relatively small and trying every possible solution would not tie up a great deal of valuable computer time. Here is a problem involving two linear equations that lends itself to a trial and error approach.

The problem is as follows: When 15 girls leave a group of boys and girls, there are two boys for every girl (lucky girls). Next, 45 boys decide to leave; then there are 5 girls for every boy (lucky boys!). How many girls were there in the group before anyone left?

Before rushing to the computer, you must recognize that this problem requires the solution of two simultaneous equations. If G equals the original number of girls and B the original number of boys, then the two equations are:

$$(G - 15) \times 2 = B$$

$$(B - 45) \times 5 = (G - 15)$$

```
   1 LET I=0
   5 PRINT "PROGRAM GROUP"
   6 PRINT
  10 FOR G=1 TO 100
  20 FOR B=1 TO 100
  25 LET I=I+1
  30 IF 2*(G-15)<>B THEN GOTO 50
  40 IF 5*(B-45)<>(G-15) THEN GO
TO 50
  45 PRINT G;" GIRLS",B;" BOYS"
  48 GOTO 90
  50 NEXT B
  60 NEXT G
  70 PRINT "NO SOLUTION"
  90 PRINT I;" COMBINATIONS TRIE
D"
```

```
PROGRAM GROUP

40 GIRLS          50 BOYS
3950 COMBINATIONS TRIED
```

The computer program uses two FOR loops (Statements 10-60 and 20-50) to try every combination of values for B and G between 1 and 100 until a solution is found or until the program runs out of values. The variable I (Statement 25) is a counter which records the number of trials required to reach a solution.

The program is straightforward and finds a solution after 3950 trials. However, it would have been a simple matter to substitute the value of B from the first equation in the second one and quickly solve the problem by hand or with the aid of a calculator. It is important to recognize that if a problem can easily be solved by other methods, the computer offers little or no advantage.

Try this problem. You may or may not want to use your computer. If Matthew can beat Jeff by one-tenth of a mile in a two-mile race and Jeff can beat Steven by one-fifth of a mile in a two-mile race, by what distance could Matthew beat Steven in a two-mile race? (Hint: the answer is not 3/10 mile.)

Brown's Books

The use of a trial and error approach can generally be improved significantly if the combinations to be tried can be narrowed down in some way. The solution to this problem illustrates how the speed of obtaining a solution can be improved well over 100 fold by combining equations and eliminating certain solution possibilities.

Here is the problem. Brown sold 48 books at a flea market, some for $3 each, some for $5 each and others for $8. He collected a total of $175. He remembered having an even number of $5 books. Can you determine how many of each kind of book he had?

The equations for solution are (letting T equal the number of $3 books, F the number of $5 books, and E the number of $8 books):

$$T + F + E = 48$$

$$3*T + 5*F + 8*E = 175$$

The first program was written simply to try all possible combinations of T, F, and E from 1 to 48. It yields three solutions for the problem, although the two solutions with an odd number of $5 books can be eliminated leaving just the one desired solution.

```
  5 PRINT "ALL POSSIBLE SOLUTION
S FOR BOOKSPROBLEM"
  6 PRINT
  7 PRINT "T";TAB 10;"F";TAB 20
;"E"
  8 PRINT
 10 FOR T=1 TO 48
 20 FOR F=1 TO 48
 30 FOR E=1 TO 48
 40 IF (T+F+E)<>48 THEN GOTO 70
 50 IF (T*3+F*5+E*8)<>175 THEN
GOTO 70
 60 PRINT T;TAB 10;F;TAB 20;E
 70 NEXT E
 80 NEXT F
 90 NEXT T
```

```
ALL POSSIBLE SOLUTIONS FOR BOOKS
PROBLEM

T           F           E

34          13          1
37          8           3
40          3           5
```

This program took approximately 22 minutes and 13 seconds to run on the Timex 1000 computer. A typical minicomputer (PDP-8/e) could run this problem in about 7.3 seconds. In either case, this is a long time to tie up the computer.

It is rather easy to combine the two equations into one by solving for T. The single equation is then:

$$2*F + 5*E = 31$$

In this equation, the limits can be reduced (from 48 used in the first run) since F cannot possibly be greater than 31/2 or 15.5 and E cannot be greater than 31/5 or 6.2. Making the appropriate program modifications leads to the second program.

```
   10
   20 FOR F=1 TO 15
   30 FOR E=1 TO 6
   40 IF (F*2+E*5)<>31 THEN GOTO
70
   50
   55 LET T=48-F-E
   90
```

```
ALL POSSIBLE SOLUTIONS FOR BOOKS
PROBLEM

T              F              E

40             3              5
37             8              3
34             13             1
```

Using this program produces a dramatic improvement in the time to solution. On the Timex, the time is approximately 2.5 seconds and on the PDP-8, about 0.16 seconds.

Since the problem states that F must be even, a final modification which steps F by two in Statement 20, can be made. This version of the program takes only 1.25 seconds to run on the Timex and 0.06 seconds to run on the PDP-8.

```
    5 PRINT "SOLUTIONS FOR BOOKS
PROBLEM WITHF = EVEN NUMBER"
   20 FOR F=2 TO 14 STEP 2
```

Notice the enormous improvement in computing time required for a solution, over 1000 fold on the Timex and 100 fold on a PDP-8. Brute force certainly is inefficient! It is generally worthwhile to think through most problems, particularly big ones, before rushing to the computer. The computer may be fast, but we just improved its performance by 1000 times by using a little common sense.

Intersection of Sets

Two sets of numbers can be combined to yield a third set by the operation of intersection. The intersection of two sets A and B is the set that contains all elements that belong to both A and B. It does not contain any other elements. The intersection is usually written A ∩ B.

For example if M = < 0,2,4,6 > and K = < 1,2,3,4 >, then M ∩ K = < 2,4 >.

This program finds the intersection of two sets of numbers. It has been written to find the intersection of the two repetitive sets described in Statements 10 and 20. In the sample run, Statement 10 describes the set

x = < 1,3,5, . . . 19 > and Statement 20 describes the set

y = < 2,5,8, . . . 29 >.

Notice that successive values of x increase by 2 and y by 3.

```
   7 PRINT "THE INTERSECTION OF
SETS X AND Y IS:"
  10 FOR X=1 TO 19 STEP 2
  20 FOR Y=2 TO 29 STEP 3
  30 IF X=Y THEN GOTO 70
  40 NEXT Y
  50 NEXT X
  60 STOP
  70 PRINT X
  80 GOTO 50
```

```
THE INTERSECTION OF SETS X AND Y
 IS:
5
11
17
```

However, if the set cannot be so neatly described, it may be desirable to rewrite the program to examine any set of data. On some computers, the Basic language includes a data statement into which series (or sets) of values can be inserted. Since the Timex 1000 does not have this statement, a different approach must by used.

In a sense, the REM statement (1) combined with Statements 8 through 40 are equivalent to a pair of DATA and READ statements on another computer. The program is set up to use the same y set as the first program, but the x set is defined in the pseudo data statement.

You should be able to see from the first combination of sets that if there is a numerical pattern in the sets which intersect, then there is also a pattern in the resulting intersecting set. In the example, the x values increase by 2 and

the y values by 3, hence the values in the intersecting set increase by 2 x 3 = 6. Although the intersection of these sets could easily be calculated by hand, the computer can be an aid in evaluating more complicated sets.

```
   1 REM 2,3,8,9,14,15,20,21,26,
27,
   4 PRINT "PROGRAM SETS-1"
   5 PRINT
   6 PRINT "THE INTERSECTION OF
SETS X AND YIS:"
   8 LET I=16514
  10 LET A$=""
  12 LET A$=A$+CHR$ PEEK I
  16 LET I=I+1
  20 IF PEEK I<>26 THEN GOTO 12
  25 LET I=I+1
  30 LET X=VAL A$
  40 FOR Y=2 TO 29 STEP 3
  50 IF X=Y THEN PRINT X
  60 NEXT Y
  65 IF PEEK I=118 THEN STOP
  70 GOTO 10
```

```
PROGRAM SETS-1

THE INTERSECTION OF SETS X AND Y
IS:
2
8
14
20
26
```

Prime Factors

A prime factor is a positive integer that has no factor except itself and one. The first ten prime factors (or numbers) are 2, 3, 5, 7, 11, 13, 17, 19, 23, and 29. The definition gives the basic method for determining whether a number is prime: divide by all smaller integers down to 2, testing whether the remainder is zero for at least one of them. If not, the number is prime.

But this is highly inefficient. It is obvious that a number is not prime if it is any even number greater than 2; hence only odd divisors need to be tried. Also, it is not necessary to try divisors greater than the square root of the number.

Since the division method is inefficient, various schemes have been devised to avoid division. The basic idea underlying all such schemes is called the sieve of Eratosthenes (276 B.C.-195 B.C.). Imagine a list of odd numbers from 3 up. Strike out every third number after 3, every fifth number after 5, and so on. This will leave only prime numbers.

```
140 DIM A(100)
150 LET C=0
160 PRINT "THIS PROGRAM WILL GI
VE THE PRIME"
170 PRINT "FACTORS OF ANY WHOLE
NUMBER."
180 PRINT "ENTER A ZERO TO STOP
THE PROGRAM"
190 PRINT
200 PRINT "WHAT IS THE NUMBER?"
210 LET X=0
220 INPUT M
225 LET M=INT M
260 PRINT
270 IF M<=0 THEN STOP
280 PRINT M.
290 LET I=1
300 LET I=I+1
310 IF I>M THEN GOTO 370
320 IF M/I<>INT (M/I) THEN GOTO
300
330 LET X=X+1
340 LET A(X)=I
350 LET M=M/I
360 GOTO 320
370 IF X=1 THEN GOTO 420
380 FOR L=1 TO X
390 PRINT A(L);" ";
400 NEXT L
410 GOTO 430
420 PRINT "IS PRIME"
430 PAUSE 4E4
440 CLS
450 GOTO 200
```

```
THIS PROGRAM WILL GIVE THE PRIME
FACTORS OF ANY WHOLE NUMBER.
ENTER A ZERO TO STOP THE PROGRAM

WHAT IS THE NUMBER?
105                3 5 7

WHAT IS THE NUMBER?
72                 2 2 2 3 3

WHAT IS THE NUMBER?
362                2 181

WHAT IS THE NUMBER?
89                 IS PRIME
```

The program here finds the prime factors of any integer, or prints out "N is prime" if the integer has no proper divisors.

Run this program for a large number of different integers and see if you can discover relationships between numbers and their prime factors. You should also try to figure out the method employed in the program to find the prime factors of any integer. To do this, you might want to draw a flowchart to show what is happening in the program. This will help you see the method used to find a prime factor and might help you in writing a program to generate primes.

In writing a program to generate prime factors, you can use the sieve method. However, as the numbers become very large, you will have to figure out a way to represent integers with more digits than your computer can handle at one time. (One approach is described on pp. 19-21 of *Computers in Mathematics.*)

Goldbach was a mathematician who made a conjecture that every even number greater than 4 can be written as the sum of two prime numbers ($16 = 11 + 5$, $30 = 17 + 13$, etc.). No one has ever proved it but no one has disproved it either. That is why it is called a conjecture. Can you write a program that will prove or disprove this conjecture? Or how about writing a program to prove Goldbach's conjecture for even numbers up to 50? You should be able to write this program with 12 or fewer statements.

Here is another problem involving prime numbers. Assume a life span of 80 years. In what year of the 20th century (1900-1999) would a person have to be born to have the maximum number of birthdays occurring in prime years? The minimum number?

Greatest Common Divisor

The greatest common divisor of a set of numbers is, as its name implies, the greatest integer that will divide into a set of two or more numbers. For example, the set of numbers 12, 20, and 28 have a greatest common divisor of 4. Nothing larger than 4 will divide evenly into all three numbers.

This program will find the greatest common divisor for any set of integers. To run it, you simply input the number of integers in your set, type them in when requested and let the program calculate the GCD. The heart of the calculation is in Statement 240.

Do you know the meaning of a relatively prime set of numbers? Can you figure out the meaning from the third sample run of the program or from runs of your own? How is a set of relatively prime numbers different from a set of prime factors? Can you find a set of 10 integers that is relatively prime?

```
   5 CLS
  40 PRINT "HOW MANY NUMBERS TO
SE"
  50 PRINT "INVESTIGATED?"
  60 INPUT N
  70 IF N=INT N THEN GOTO 100
  80 PRINT "TRY AGAIN"
  90 GOTO 60
 100 DIM X(N)
 105 PRINT N
 110 PRINT "INPUT NUMBERS"
 120 LET S=1E25
 130 FOR K=1 TO N
 140 INPUT X(K)
 150 IF X(K)>S THEN GOTO 180
 160 LET S=X(K)
 180 PRINT X(K);" ";
 190 NEXT K
 200 PRINT
 210 LET G=0
 220 FOR M=2 TO S
 230 FOR I=1 TO N
 240 IF X(I)/M<>INT (X(I)/M) THE
N GOTO 270
 250 NEXT I
 260 LET G=M
 270 NEXT M
 280 PRINT "THE NUMBERS ";
 290 IF G>0 THEN GOTO 320
 300 PRINT "ARE RELATIVELY PRIME
"
 310 GOTO 330
 320 PRINT "HAVE THE G.C.D. ";G
 330 PRINT
 340 PRINT "ANOTHER SET?"
 350 INPUT A$
 360 IF CODE A$=62 THEN RUN
```

```
HOW MANY NUMBERS TO BE
INVESTIGATED?
3
INPUT NUMBERS
12 36 96
THE NUMBERS HAVE THE G.C.D. 12

ANOTHER SET?


HOW MANY NUMBERS TO BE
INVESTIGATED?
3
INPUT NUMBERS
20 36 96
THE NUMBERS HAVE THE G.C.D. 4

ANOTHER SET?


HOW MANY NUMBERS TO BE
INVESTIGATED?
3
INPUT NUMBERS
20 36 97
THE NUMBERS ARE RELATIVELY PRIME

ANOTHER SET?
```

Cryptarithmetic Problems

Cryptarithmetic or alphametic problems are arithmetic expressions in which the digits are replaced by letters of the alphabet. Each digit is associated with a letter to produce an interesting statement, for example:

```
  SEND
+ MORE
-------
 MONEY
```

If the college student who sent this message to his father needed $106.52 for plane fare home, this was the right message to send since this combination of letters has one unique solution, in particular:

```
  9567
+ 1085
-------
 10652
```

However, if the student let things go to the last moment and was in more of a rush, he might have reworded the message:

```
  WIRE
+ MORE
-------
 MONEY
```

In this case, how much should his dad send? Earlier in the book, trial and error approaches to solving problems were discussed. It was noted that the brute force approach of trying every alternative was sometimes appropriate. Is it in this case?

No! The number of possible alternative solutions is the factorial of the number of different letters in the alphametic expression, i.e., 8! or 40,320. A program to try out every one of these possibilities would run for a l-o-n-g time.

In this case it is much more efficient to apply some common sense to narrow down the number of alternatives. The best approach to this process is to divide up the search space into large classes (or sets), according to a common property shared by members of each class, and then attempt to eliminate entire classes by the method of contradiction.

Consider the "WIRE + MORE = MONEY" problem. Can E = 0? Since E + E = Y, Y must also equal 0, contradicting the fact that Y and E must be different digits. Thus, the entire class of solutions in which E = 0 can be ruled out.

Consider E = 3. Now Y = 6 and there is no carry to the next column. So in this column R + R = E or E + 10 if a carry is involved. But in either case E must be an even number since 2R is always even; this contradicts the assumption that E = 3.

By following this type of classificatory contradiction process for each of the

digits in the order E, R, I, O and N, the computer program will search out all possible solutions to the problem. Unlike the "send more money" problem, the "wire more money" problem has five possible solutions. A smart father would choose the lowest solution and wire his son $103.48.

```
   5 PRINT "THIS PROGRAM SEARCHE
S OUT ALL"
  10 PRINT "SOLUTIONS TO THE CRY
PTARITHMETIC"
  20 PRINT "PROBLEM."
  30 PRINT
  40 PRINT "     W    I    R    E"
  50 PRINT "   + M    O    R    E"
  60 PRINT "-------------------"
  70 PRINT "M    O    N    E    Y"
  80 PRINT
  90 LET M=1
 100 FOR E=2 TO 9
 110 LET Y=E+E
 120 IF Y>=10 THEN GOTO 150
 130 LET C1=0
 140 GOTO 170
 150 LET C1=1
 160 LET Y=Y-10
 170 FOR R=0 TO 9
 180 IF R=M OR R=E OR R=Y THEN G
OTO 470
 190 IF R+R+C1=E THEN GOTO 220
 200 IF R+R+C1=E+10 THEN GOTO 24
0
 210 GOTO 470
 220 LET C2=0
 230 GOTO 250
 240 LET C2=1
 250 FOR I=0 TO 9
 260 IF I=M OR I=E OR I=Y OR I=R
 THEN GOTO 460
 270 FOR O=0 TO 9
 280 IF O=M OR O=E OR O=Y OR O=R
 OR O=I THEN GOTO 450
 290 LET N=I+O+C2
 300 IF N>=10 THEN GOTO 330
 310 LET C3=0
 320 GOTO 350
 330 LET C3=1
 340 LET N=N-10
 350 IF N=M OR N=E OR N=Y OR N=R
 OR N=I OR N=O THEN GOTO 450
 360 FOR W=0 TO 9
 370 IF W=M OR W=E OR W=Y OR W=R
 OR W=I OR W=O OR W=N THEN GOTO
 440
 380 IF O+10<>W+M+C3 THEN GOTO 4
40
 390 PRINT "  ";W;I;R;E
 400 PRINT "  ";M;O;R;E
 410 PRINT "-----"
 420 PRINT M;O;N;E;Y
```

```
430 GOSUB 500
440 NEXT W
450 NEXT O
460 NEXT I
470 NEXT R
480 NEXT E
490 STOP
500 PAUSE 4E4
510 CLS
520 RETURN
```

```
THIS PROGRAM SEARCHES OUT ALL
SOLUTIONS TO THE CRYPTARITHMETIC
PROBLEM.

    W   I   R   E
  + M   O   R   E
---------------
M   O   N   E   Y

 9762
 1062
-----
10824

 9274
 1074
-----
10348

 9574
 1074
-----
10648

 9287
 1087
-----
10374

 9587
 1087
-----
10674
```

There are other approaches to solving cryptarithmetic problems, but all of them benefit greatly from reducing the search space as much as possible before putting the problem on the computer. See if you can devise another successful approach and write a program to implement it.

Here are some problems for you to try.

```
  DONALD
+ GERALD
--------
  ROBERT
```

```
  TWO
x TWO
-----
 THREE
```

```
  ABCDE
x     4
-------
  EDCBA
```

```
  ABC
x DE
----
  FEC
 DEC
----
 HGBC
```

```
    THE
  EARTH
  VENUS
 SATURN
+ URANUS
--------
 NEPTUNE
```

```
  SPRING
   RAINS
   BRING
+ GREEN
-------
  PLAINS
```

VIOLIN + VIOLIN + VIOLA + CELLO = QUARTET

THREE + NINE = EIGHT + FOUR

```
   ONE
   TWO
+ FIVE
------
 EIGHT
```

```
  FORTY
    TEN
+   TEN
-------
  SIXTY
```

```
  FIVE
-FOUR
-----
  ONE
+ ONE
-----
  TWO
```

Sailors and Monkey Problem

There are many variations of the sailors and monkey problem. Here is one of them.

Five sailors and a monkey were on an island. One evening the sailors rounded up all the coconuts they could find and put them in a large pile. Being exhausted from working so hard, they decided to wait and divide them up equally in the morning. During the night, a sailor awoke and separated the nuts into five equal piles, but had one nut left over which he gave to the monkey. He took one pile, hid it, and pushed the other four together and went back to sleep. He was followed in this action by the other four sailors, each of whom did exactly the same thing. Next morning the remaining nuts were divided equally with one remaining nut going to the monkey. What is the smallest number of coconuts with which they could have begun?

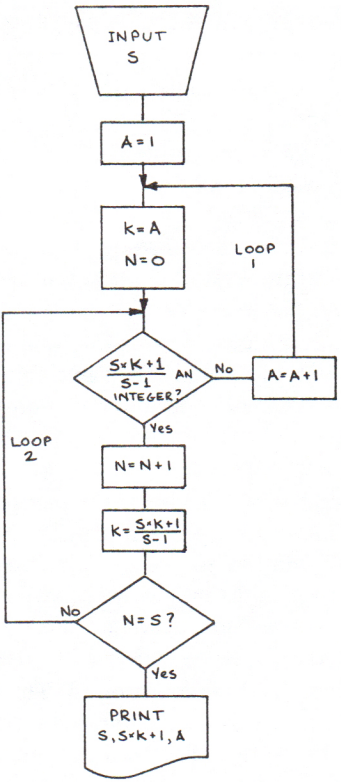
Although there is an elegant algebraic solution to this problem, a more suitable approach for the computer is that of working backwards. A typical solution to a problem can be thought of as a path that leads from the given information to the goal. However, in this case the goal, or final state, is known, thus it is easier to start there and work backwards to the initial state.

As mentioned at the outset, many sailor and monkey problems exist, in fact, an infinite number of them. For example, instead of five sailors, there could be three or six or 14. Thus it is desirable to devise a general solution instead of just one to solve one specific problem.

In the flowchart and computer program, S is the number of sailors and A is the number of coconuts that each sailor received in the final division of the pile. Since one coconut was given to the monkey at each division, the total number of coconuts left in the morning must be S x A + 1. But this pile came from pushing together S - 1 equal piles. Thus, the key condition that must hold for (S x A + 1) / (S - 1) to be an integer K, which represents the number of coconuts that the last sailor stole from a pile of S x K + 1 coconuts. But this pile is the result of pushing together S - 1 equal piles by the previous thief, so again (S x K + 1) / (S - 1) is an integer and so on back through all S raids on the pile.

Note in the flowchart (and program) that the first trial value for A is 1 (Statement 160). In Statement 200 this value is increased by 1 until the value of (S x K + 1)/ (S - 1) is an integer as tested for in Statement 190. This process is then continued until the counter for the second loop, N (nighttime pile divisions) equals the number of sailors.

Although the program will work for any number of sailors, it takes a fairly long time to run for more than five. Remembering what you have learned earlier in this chapter, can you devise a way to make the program more efficient?


INPUT
S
A=1
K=A
N=0
LOOP
1
S*K+1 / S-1 AN INTEGER?
No
A=A+1
Yes
LOOP
2
N=N+1
K=S*K+1 / S-1
No
N=S?
Yes
PRINT
S, S*K+1, A

```
 100 PRINT "THIS PROGRAM SOLVES
THE SAILORS"
 105 PRINT "AND MONKEY PROBLEM B
Y WORKING"
 110 PRINT "BACKWARDS."
 120 PRINT
 130 PRINT "HOW MANY SAILORS ARE
 ON THE"
 135 PRINT "ISLAND? ";
 140 INPUT S
 145 PRINT S
 150 PRINT
 160 LET A=1
 170 LET K=A
 180 LET N=0
 190 IF (S*K+1)/(S-1)=INT ((S*K+
1)/(S-1)) THEN GOTO 220
 200 LET A=A+1
 210 GOTO 170
 220 LET N=N+1
 230 LET K=(S*K+1)/(S-1)
 240 IF N=S THEN GOTO 260
 250 GOTO 190
 260 PRINT "THE LEAST NUMBER OF
COCONUTS"
 270 PRINT "THAT ";S;" SAILORS C
AN BEGIN WITH"
 280 PRINT "IS ";S*K+1
 290 PRINT
 300 PRINT "IN THE MORNING, EACH
 SAILOR GETS"
 310 PRINT A
```

```
THIS PROGRAM SOLVES THE SAILORS
AND MONKEY PROBLEM BY WORKING
BACKWARDS.

HOW MANY SAILORS ARE ON THE
ISLAND? 5

THE LEAST NUMBER OF COCONUTS
THAT 5 SAILORS CAN BEGIN WITH
IS 15621

IN THE MORNING, EACH SAILOR GETS
1023
```

Palindromes

A palindrome is a word, verse, or number that reads the same backwards or forwards. For example, the words "mom" and "eye" are palindromes. So are each of the lines in this verse:

Egad, a base life defiles a bad age
Doom an evil deed, liven a mood
Harass sensuousness, Sarah
Golf; No, sir, prefer prison-flog
Ban campus motto, "Bottoms up, MacNab"

Numeric palindromes are those numbers which read the same backward as forward. The examination of these numbers is a field rich with possibilities for creative computing.

One conjecture concerning palindromes raises an interesting unanswered question. Begin with any positive integer. If it is not a palindrome, reverse its digits and add the two numbers. If the sum is not a palindrome, treat it as the original number and continue. The process stops when a palindrome is obtained. For example, beginning with 78:

```
   78
+  87
-----
  165
+561
-----
  726
+ 627
-----
 1353
+3531
-----
 4884
```

The conjecture, often assumed true, is that this process will always lead to a palindrome. And indeed that is just what usually happens. Most numbers less than 10,000 will produce a palindrome in less than 24 additions. But there is a real thorn in the side of this conjecture, the number 196. Can you determine if a palindrome will ever be produced with a starting number of 196?

The number 196 will produce 1675 after two reversals, but after 100 reversals the resultant sum has 47 digits and is still not palindromic. Why mention 1675? Because ten other numbers under 1000 will also lead to the sum of 1675 and thus may not become palindromic. The first five of these numbers are 196, 295, 394, 493, and 592. What are the other five?

The program here will accept any number as a starting value and complete the process of adding the successive reversals and testing if the sum is a palindrome. Try it with some numbers and see if you can identify any patterns. If the screen fills, press CONT to clear the screen and continue the calculation.

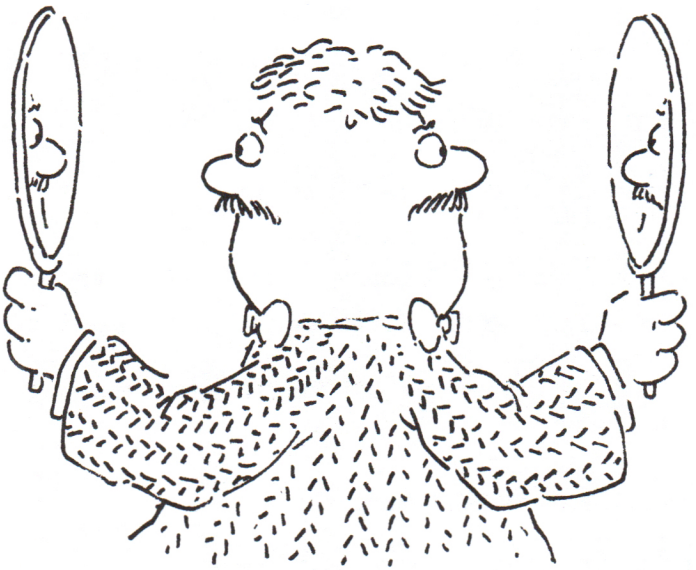
```
  1 PAUSE 300
  5 DIM B(50)
 10 CLS
 20 PRINT "GIVE ME A NUMBER"
 30 INPUT A
 40 PRINT
 50 LET B=0
 60 LET B=B+1
 70 LET A=A/10
 80 IF INT A>0 THEN GOTO 60
 90 FOR C=B TO 1 STEP -1
100 LET A=A*10
110 LET B(C)=INT (A-10*INT (A/1
0))
120 NEXT C
130 LET D=0
140 FOR C=1 TO INT (B/2)
150 IF B(C)<>B(B+1-C) THEN LET
D=1
160 NEXT C
170 FOR C=B TO 1 STEP -1
180 PRINT CHR$ (B(C)+28);
190 NEXT C
200 IF D=1 THEN GOTO 230
210 PRINT " IS A PALINDROME."
220 RUN
230 PRINT " IS NOT A PALINDROME
."
240 IF B/2>INT (B/2) THEN LET B
(INT (B/2)+1)=2*B(INT (B/2)+1)
250 FOR C=1 TO INT (B/2)
260 LET B(C)=B(C)+B(B+1-C)
270 NEXT C
280 FOR C=1 TO INT (B/2)
290 LET B(B+1-C)=B(C)
300 NEXT C
310 LET B(B+1)=0
320 FOR C=1 TO B
330 LET B(C+1)=B(C+1)+INT (B(C)
/10)
340 LET B(C)=B(C)-10*INT (B(C)/
10)
350 NEXT C
370 IF B(B+1)>0 THEN LET B=B+1
380 GOTO 130
```

```
GIVE ME A NUMBER

19 IS NOT A PALINDROME.
110 IS NOT A PALINDROME.
121 IS A PALINDROME.
```

```
GIVE ME A NUMBER

79 IS NOT A PALINDROME.
176 IS NOT A PALINDROME.
847 IS NOT A PALINDROME.
1595 IS NOT A PALINDROME.
7546 IS NOT A PALINDROME.
14003 IS NOT A PALINDROME.
44044 IS A PALINDROME.


GIVE ME A NUMBER

96 IS NOT A PALINDROME.
165 IS NOT A PALINDROME.
726 IS NOT A PALINDROME.
1353 IS NOT A PALINDROME.
4884 IS A PALINDROME.
```

Using this method, write a program that examines all the integers between 1 and 10,000 excluding those that sum to 1675 at any point. What does this show? By the way, you will have to devise a way to deal with 14-digit integers which are larger than your computer can normally handle.

Huh? Is this program "too hot to hoot?"



4

Convergence and Recursion

The computer is especially suitable for doing repetitive and tedious calculations. Two mathematical approaches for solving problems that involve repetitive calculations are convergence and recursion.

Some problems can be reasonably easily stated in words or described with a few simple equations but there are many possible solutions. For example, how many ways can you make change for a dime? It is simply stated and the number of ways can be enumerated fairly easily: two nickels, one nickel and five pennies, or ten pennies, three ways in all. But if you want to solve for all the ways of making change for a dollar or five dollars, it would be nice to have some help.

Help on this kind of problem comes from a class of computer program that simply breaks the problem into smaller ones and counts up all the alternative solutions according to a set of rules. But an even more powerful technique is known as recursion. Using this technique, a simple solving algorithm or routine is set up to solve the smallest subset of the problem. The unique power in a recursive routine comes from the ability of the routine actuallly being able to call itself. This is discussed further in the second program in this section.

Another approach for solving problems that do not have an exact answer is that of successive approximations. For example, the exact value of pi, *e* or the length of an irregular curve cannot be precisely determined. But by means of increasingly accurate approximations, it is possible to approach the desired value from above or below or to converge on it from two directions. The last four programs in this chapter illustrate successive approximations and convergence.

Change For a Dollar

Even though there is very little you can buy for a penny these days, the coin will probably be around for some time to come since it is needed to make change for odd amounts of sales tax and to fill up penny collections.

Today U.S. coinage consists of five coins: penny, nickel, dime, quarter, and half dollar. How many ways can coins of these denominations be used to make change for one dollar? For example, one way is two half dollars, another is one half dollar and two quarters, and so on. Make a best guess now and write it down before you read further.

There are several different ways to approach a problem of this kind. One is to break it down into smaller, more easily solved problems. In other words, how many ways can you make change for a quarter? For a dime? You would solve these subproblems and combine the answers to give the overall solution.

If you were more mathematically inclined, you could write a series of equations relating each piece of change to every other one and to the dollar and solve them.

A third approach is to do the problem by writing down combinations until all the different possibilities are exhausted (or until you are exhausted) and then count them all up. This might be called solving the problem by exhaustion and is a method quite suitable for putting on the computer.

Write a program that uses this approach to solve the problem. If you use loops and count by one, it could take a long time for the computer to run through all the possible combinations, possibly many hours.

Also, if you want to print out all the possible combinations, be warned that the printing could also take quite some time and a fair amount of paper. There are more combinations than you might think!

In fact, most people will not be able to guess the answer to this problem, or even come close. Ask several of your friends how many ways they think a dollar can be changed. Record all the responses and then tabulate them on your computer. What is the mean (average) of all the guesses? The extremes?

The program included here uses the first method discussed to solve the problem, in particular, breaking down the problem into subproblems and then combining the solutions into one final answer.

First, the main problem is broken into the next smaller one of making change for half dollars. There are three such problems: no half dollars (H = 0), one half dollar (H = 1), and two half dollars (H = 2). The last problem is trivial since there is only one way, but the other two need to be broken down further.

This is done by dividing the remaining money into quarters and considering the subproblems on down to the lower denominations. As the number of subproblems is expanded, each one becomes easier to solve. In fact,

subgoals, which can be solved in only one way, are finally reached. For example, if H = 1, Q = 1, D = 2, and N = 0, then the pennies (P) must equal five in order that the total equal 100.

Notice that at the quarter, dime and nickel stages, adjustments are made in the limits of the loops depending upon how much money there is left to change. For example, if H = 1, the only possible subgoals for quarters are 0, 1, and 2, but not 3 or 4. Also notice that there is no need to test combinations of coins to see if they add up to 100, nor is it necessary to include the penny as a variable. Simply counting the number of subgoals is sufficient since each one can be solved in only one way.

```
 10 REM PROGRAM DOLLAR COMOUTES
 15 REM THE NUMBER OF WAYS TO
 20 REM MAKE CHANGE FOR $1
 30 LET C=0
 40 FOR H=0 TO 2
 50 FOR Q=0 TO 4-2*H
 60 FOR D=0 TO 10-5*H-.5*Q
 70 FOR N=0 TO 20-10*H-5*Q-2*D
 80 LET C=C+1
 90 NEXT N
100 NEXT D
110 NEXT Q
120 NEXT H
130 PRINT "THERE ARE ";C;" DIFF
ERENT WAYS"
140 PRINT "TO CHANGE A DOLLAR B
ILL."
```

```
THERE ARE 292 DIFFERENT WAYS
TO CHANGE A DOLLAR BILL.
```

Try to make some changes in this program or write a new one to solve the following problems. Say you want two quarters in your change to play some video games. In how many ways can a dollar be changed to provide at least two quarters?

Visiting a small town, you find the parking meters still take pennies. In how many ways might you get change so that you had at least three pennies? Is this any different than the number of ways that would give you five pennies? Say you want to make a phone call also; in how many ways can you change a dollar to produce at least four pennies and one dime?

Change For Any Amount to $5.00

Another way to attack the change problem in the previous section is by means of the programming technique called recursion. Get familiar with this one—it is very powerful! Donald Piele and Larry Wood described this method in an issue of *Creative Computing.*

First, define the variables which represent the number of ways to make change for n cents using the coins specified:

A Only pennies
B Nickels and pennies
C Dimes, nickels, and pennies
D Quarters, dimes, nickels, and pennies
E Halves, quarters, dimes, nickels, and pennies.

Initially, there are two subproblems in making change for n cents. In the first, no half dollars are used, and D is the number of ways to change n cents. Second, when one or more halves are used, after one is paid, there remain n–50 cents to pay which can be done in E_{n-50} ways.

Since these two cases are mutually exclusive, it can be inferred that $E_n = D_n + E_{n-50}$. Similarly,

$D_n = C_n + D_{n-25}$
$C_n = B_n + C_{n-10}$
$B_n = A_n + B_{n-5}$

Now, begin with the simplest case and build up to E_{100}. First of all, it is easy to understand why $E_0 = 1$. From above, when $n = 50$, $E_{50} = D_{50} + E_0$, and it is possible to make change for 50 cents only one more way if half dollars are allowed. Therefore $E_0 = 1$. Likewise, $D_0 = C_0 = B_0 = A_0 = 1$. It is also true that $A_n = 1$ for all values of n since there is only one way to make change using only pennies. Now the recursive relationships can be used to solve the original problem.

This is the strategy used in the program. It also has the added advantage that it can count the number of ways of making change (with coins) for any specified amount.

Now it is your turn. Can you modify the program here to include dollar bills so it could count the number of ways to make change for any amount up to $10.00?

Using any method of change making you prefer, write a program to make change for one ruble. Russian coins come in denominations of 1, 2, 3, 5, 10, 15, 20, 50, and 100 kopecks. There are 100 kopecks in one ruble.

```
    5 PRINT "PROGRAM CHANGE COMPU
TES THE"
   10 PRINT "NUMBER OF WAYS OF MA
KING CHANGE"
   15 PRINT "IN COINS FOR ANY AMO
UNT OF MONEYUP TO $5.00."
   25 PRINT
   30 DIM A(101)
   35 DIM B(101)
   40 DIM C(101)
   45 DIM D(101)
   50 DIM E(101)
   55 PRINT "HOW MUCH TO CHANGE?"
   60 INPUT C
   65 LET M=INT (20*C)+1
   70 LET A(1)=1
   75 LET B(1)=1
   80 LET C(1)=1
   85 LET D(1)=1
   90 LET E(1)=1
  100 FOR J=2 TO M
  105 LET A(J)=1
  110 LET B(J)=A(J)+B(J-1)
  115 LET C(J)=B(J)
  120 IF J<=2 THEN GOTO 130
  125 LET C(J)=B(J)+C(J-2)
  130 LET D(J)=C(J)
  135 IF J<=5 THEN GOTO 145
  140 LET D(J)=C(J)+D(J-5)
  145 LET E(J)=D(J)
  150 IF J<=10 THEN GOTO 160
  155 LET E(J)=D(J)+E(J-10)
  160 NEXT J
  170 PRINT
  175 PRINT "YOU CAN MAKE CHANGE
FOR $";C
  180 PRINT "IN ";E(M);" DIFFEREN
T WAYS."
```

```
PROGRAM CHANGE COMPUTES THE
NUMBER OF WAYS OF MAKING CHANGE
IN COINS FOR ANY AMOUNT OF MONEY
UP TO $5.00.

HOW MUCH TO CHANGE?

YOU CAN MAKE CHANGE FOR $5
IN 59576 DIFFERENT WAYS.
```

```
PROGRAM CHANGE COMPUTES THE
NUMBER OF WAYS OF MAKING CHANGE
IN COINS FOR ANY AMOUNT OF MONEY
UP TO $5.00.

HOW MUCH TO CHANGE?

YOU CAN MAKE CHANGE FOR $3.56
IN 17320 DIFFERENT WAYS.
```

Perhaps the most famous problem used to demonstrate the principles of recursion is the Towers of Brahma. It is sometimes called the Towers of Hanoi or Pharoah's Needles. Here is the problem in as close to original form as possible. You should be able to solve it with a relatively short program using recursion.

In the great temple at Benares beneath the dome which marks the center of the world rests a brass plate in which are fixed three diamond needles, each a cubit high and as thick as the body of a bee. On one of these needles, at the Creation, God placed 64 discs of pure gold, the largest disc resting on the brass plate and the others getting smaller and smaller up to the top one. This is the Tower of Brahma.

Day and night unceasingly, the priests transfer the discs from one needle to another, according to the fixed and immutable laws of Brahma. These laws require that the priest on duty must not move more than one disc at a time and that he must place this disc on a needle so there is no smaller disc below it. When the 64 discs shall have been thus transferred from the needle which, at the Creation, God placed them, to one of the other needles; tower, temple, and Brahmans alike will crumble into dust, and with a thunderclap, the world will vanish.

If the priests were to effect one transfer every second, and work 24 hours per day for each day of the year, it would take them 58,454,204,609 decades plus slightly more than six years to perform the feat, assuming they never made a mistake—for one small slip would undo all the work.

How many transfers are required to fulfill the prophecy? Try out your program with fewer discs than 64 to make sure you are on the right track. Here is a table of the first few transfers:

Discs	*Moves*	*Discs*	*Moves*
1	1	6	63
2	3	7	127
3	7	8	255
4	15	9	511
5	31	10	1023

Converge on e and Pi

An incredibly important mathematical constant is designated by the small letter *e*. This constant is both irrational and transcendental. Look up those terms in a dictionary or math book if you wish, or just plunge on to the next paragraph.

The constant *e* was first derived by John Napier, also the inventor of logarithms, to whom we owe an eternal debt of gratitude. Why? If *e* had never been discovered, advances in mathematics, physics, and astronomy would have lagged a century or more, because *e* is the base of all natural logarithms and these logarithms are the basis for many branches of science and mathematics.

How is *e* calculated? The constant *e* is the limiting value of this expression as n approaches infinity:

$$e = (1+1/n)^n$$

Its exact value can never be found, but to 15 places *e* equals 2.718281828459045 . . . How can *e* be calculated? First take 1-½ and square it; that equals 2-¼. Then cube 1-⅓ and you get 2.3686. Raising 1-¼ to the fourth power gives 2.414, and so on. Write a program for this method and have it print out the initial value of *e* and each value after each additional fraction is added on.

Another approach is to expand the expression above using the binominal theorem and, again, letting n approach infinity. The expression for the expansion is:

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} \cdots \frac{1}{n!}$$

Now here is where the computer can again be of some assistance. Since 3! is 3*2! and 4! is 4*3!, all the calculations need not be done for each additional fraction. Look at the program and particularly note the calculations in Statements 150 and 170.

```
100 LET E=1
110 LET I=0
120 LET I=I+1
130 LET D=1
140 FOR J=1 TO I
150 LET D=D*J
160 NEXT J
170 LET E=E+1/D
180 SCROLL
190 PRINT E
200 GOTO 120
```

Note: Run this program in Slow mode

```
 2
 2.5
 2.6666667
 2.7083333
 2.7166667
 2.7180556
 2.718254
 2.7182788
 2.7182815
 2.7182818
 2.7182818
 2.7182818
```

Pi is another important mathematical constant that is irrational (meaning its exact value can never be determined) and transcendental (meaning it is not the solution to any algebraic equation). Interestingly, pi was known and used by the ancients. Archimedes, who lived in the second century B.C., by using a regular polygon of 96 sides (nearly a circle), proved that the value of pi was less than 22/7 and greater than $3\text{-}^{10}/_{71}$, a remarkable achievement for the mathematics of his day.

Ptolemy in 150 A.D. used the value of 3.1416 for pi and in the middle of the sixteenth century the amazing fraction 355/113 was discovered, giving the value of pi accurately to six decimal places.

Incidentally, in 1897, the General Assembly of Indiana passed a bill ruling that the value of pi was four.

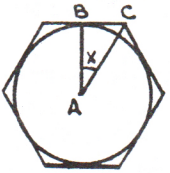
Several infinite series can be used to grind out increasingly accurate values for pi. One such series is $(1\text{-}1/3 + 1/5 - 1/7 + 1/9 \ldots)$. This series is called an arithmetic series and converges very slowly. The program here displays only every 500th value of the series. Don't be alarmed if the program does not seem to be running very fast; a fair amount of calculating is going on between each value that is printed.

```
300 LET S=1
310 LET I=1
320 LET Q=0
325 LET P=0
330 LET Q=Q+1
335 LET P=P+S/I
340 LET I=I+2
350 LET S=-S
360 IF Q<499 THEN GOTO 330
370 LET Q=0
375 SCROLL
380 PRINT P*4
385 PAUSE 150
390 GOTO 330
```

SAMPLE RUN

```
3.1435967
3.1405907
3.1422607
3.1410917
3.1419935
3.1412587
3.141879
3.1413422
3.1418153
```

Actually, the approach used by Archimedes converges much more quickly and, with the aid of a computer, it is possible to go far beyond the 96-sided polygon used by Archimedes.



His approach was to construct inscribed and circumscribed polygons and measure the perimeters to approximate the circumference of a circle. Consider a polygon circumscribed around a circle of radius 1. The perimeter equals the length of one side times the number of sides. Since the tangent of x = AB/BC, but BC = 1, then tan (x) = AB and the length of a side = 2 tan (x). Since the circumference = 2 pi r and r = 1, then pi is the circumference (or perimeter of an n-sided polygon) divided by 2.

Similar trigonometry leads to the perimeter of an inscribed polygon being equal to the number of sides times sin (x) * cos (x).

The second program produces values for pi using inscribed and circumscribed polygons. Unfortunately, there is one large flaw in the program, because degrees must be converted into radians in Statement 30. This means, of course, that you must already know the value of pi, since the conversion factor is 360 degrees divided by 2 pi.

Setting this flaw aside, it is interesting to note how quickly this program converges on the value of pi compared to the preceding one. That is because this one converges geometrically rather than arithmetically.

Can you figure out a geometric convergence to the value of pi that does not require that you know it (or a conversion factor) before you start?

```
10 LET N=6
20 LET N=2*N
30 LET X=360/(N*57.29578)
40 SCROLL
50 PRINT N*SIN X*COS X/2,N*TAN
(X)/2
60 GOTO 20
```

```
SAMPLE RUN

2.5980762       3.4641016
3               3.2153903
3.1058285       3.1596599
3.1326286       3.1460862
3.1393502       3.1427146
3.1410319       3.141873
3.1414525       3.1416627
3.1415576       3.1416102
3.1415839       3.141597
3.1415904       3.1415937
```

Convergence on Pi Revisited

In answer to the question posed in the last paragraph of the preceding section, here is a way to converge on pi without knowing its value beforehand.

As in the previous program, the basic approach is to add up the length of the sides on an inscribed polygon and divide by 2r to obtain a value for pi. The program starts with a square (four sides) and doubles the number of sides each time. If the old side length is S, then the length of S' of a side of a new polygon with twice as many sides is obtained by applying the Pythagorean theorem. In particular,

$$X^2 + (S/2)^2 = R^2$$

$$(R-X)^2 + (S/2)^2 = (S')^2$$

Thus,

$$S' = \sqrt{(R-\sqrt{R^2 - (S/2)^2})^2 + (S/2)^2}$$

It is easy to reduce this formula algebraically, but accuracy suffers if this is done. Also, you will find that S*S is slightly more accurate than S**2.

Unfortunately, striving for maximum accuracy is somewhat moot on a computer that does not have double precision arithmetic. Notice that accuracy does not improve with more than 8192 sides and, indeed, as numbers in the calculations start to exceed the capacity of the computer (4194304 sides), the accuracy starts to deteriorate badly.

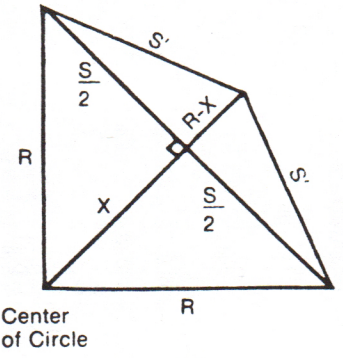
There is yet another method to compute pi by convergence. It uses discoveries of Gregory and Euler. Gregory discovered the formula for arctangent:

$$\text{Arctan } x = x - x^3/3 + x^5/5 - x^7/7 + \ldots$$

Euler came up with a rather interesting formula for pi:

$$\text{pi} = 4\,(\arctan(1/2) + \arctan(1/3))$$

See if you can combine these two formulas to calculate pi. If you are very clever, you can do your calculation to yield far more than the seven decimal place accuracy obtained by the programs presented here so far.



```
  10 PRINT "THE MACHINE VALUE FO
R PI IS"
  20 PRINT PI
  30 LET R=VAL "10"
  40 PRINT
  50 PRINT "SIDES","PERIMETER"
  60 PRINT
  70 LET Z=2
  80 LET N=Z+Z
  90 LET S=R*SQR Z
 100 FOR K=SGN PI TO VAL "28"
 110 IF K=14 THEN PAUSE 4E4
 120 IF K=14 THEN CLS
 130 PRINT N,(S*N)/(Z*R)
 140 LET Y=S*S/(Z+Z)
 150 LET X=R-SQR (R*R-Y)
 160 LET S=SQR (X*X+Y)
 170 LET N=Z*N
 180 NEXT K
```

```
THE MACHINE VALUE FOR PI IS
3.1415927

SIDES                PERIMETER

4                    2.8284271
8                    3.0614675
16                   3.1214452
32                   3.1365485
64                   3.1403312
128                  3.1412773
256                  3.1415138
512                  3.141573
1024                 3.1415877
2048                 3.1415914
4096                 3.1415924
8192                 3.1415926
16384                3.1415927

(SECOND SCREEN)

32768                3.1415927
65536                3.1415927
131072               3.1415927
262144               3.1415927
524288               3.1415927
1048576              3.1415927
2097152              3.1415927
4194304              3.1415928
8388608              3.1415932
16777216             3.1415947
33554432             3.1416001
67108864             3.1416258
134217730            3.1417253
268435460            3.1421231
536870910            3.143714
```

Length of Any Curve

The previous programs have demonstrated how it is possible to compute a very accurate value of pi by adding together the length of the sides of a polygon as it approaches a circle and dividing by 2r.

Using a similar approach, it should be possible to inscribe a polygon, or portions of a polygon, inside any regular curve and thus determine the length of the curve. This program approximates the length of any curve as defined in Statement 180 by dividing it into an increasing number of subintervals and computing the sum of the secants (a straight line that cuts a curve at two or more points).

To run the program, you must enter the formula or equation describing your curve in Statement 180 in the form:

180 LET A$ ="(your function of x)"

Note that the function must be enclosed in quotation marks. You then type RUN and enter the end points of the curve you want to use in your calculation. These points are entered in the form of the abscissa, which means the horizontal (or y) coordinate of the point.

The program is written to sum the successive secant lengths and to calculate the percent of change in each summation compared to the preceding one. The sample run uses the function $2x^3 + 3x^2 - 2x + 3$. Note the substantial improvements in the length calculation as the number of intervals increases from 2 to 16, but the rather slight improvements beyond 16.

Try this program with different curves and functions. It might help to plot the function first (remember the program to do that?) and then compute its length. Is this method of length calculation more accurate for a function with no changes in direction of the curve within the interval selected or for a function with one or more changes? Why?

```
  10 PRINT "WHAT ARE THE ABSCISS
AS OF THE"
  20 PRINT "END POINTS OF THE IN
TERVAL WHOSE"
  40 PRINT "LENGTH YOU WANT (SMA
LLER ONE"
  50 PRINT "FIRST: P,Q)"
  60 INPUT P
  70 INPUT Q
  80 IF P<Q THEN GOTO 110
  90 PRINT "P MUST BE LESS THAN
Q"
 100 GOTO 60
```

```
 110 PRINT P;"    ";0
 115 PRINT
 120 PRINT "NO. OF    SUM OF
PERCENT"
 130 PRINT "SUBIN-    SECANT
CHANGE IN"
 140 PRINT "TERVALS   LENGTHS
LENGTH"
 150 PRINT "-------   -------
----------"
 155 PRINT
 160 LET S1=0
 170 FOR N=1 TO 9
 180 LET A$="2*(X*X*X)+3*(X*X)-2
*X+3"
 190 LET E=2**(N-1)
 200 LET H=(Q-P)/E
 210 LET S=0
 220 FOR I=0 TO E-1
 230 LET X=P+I*H+H
 240 LET Y=VAL A$
 250 LET X=P+I*H
 260 LET Z=VAL A$
 270 LET L=SQR ((Y-Z)*(Y-Z)+H*H)
 280 LET S=S+L
 290 NEXT I
 300 IF S1>0 THEN GOTO 330
 310 PRINT E;TAB 9;S;TAB 20;"NO
PREV VAL"
 320 GOTO 350
 330 LET P5=((ABS (S1-S))/S1)*10
0
 340 PRINT E;TAB 9;S;TAB 20;P5
 350 LET S1=S
 360 NEXT N


WHAT ARE THE ABSCISSAS OF THE
END POINTS OF THE INTERVAL WHOSE
LENGTH YOU WANT (SMALLER ONE
FIRST: P,Q)
-1    6

NO. OF    SUM OF       PERCENT
SUBIN-    SECANT       CHANGE IN
TERVALS   LENGTHS      LENGTH
-------   -------      ---------

1         525.04667    NO PREV VAL
2         525.15826    .021254739
4         529.65225    0.8557393
8         531.01713    0.25769475
16        531.96426    0.17836008
32        532.01657    .0098333207
64        532.04168    .004720272
128       532.04856    .0012937676
256       532.05027    .00032138759
```

Converge on a Square Root

Since most square roots are irrational, methods used to calculate them usually involve successive approximations. Although you can simply call up the square root function in Basic or on many pocket calculators, it is interesting to explore various methods of calculating square roots without these built-in functions. After all, these built-in functions are nothing more than successive approximation routines already installed in the machine.

Obviously, a square root is the inverse of the operation of squaring a number. All of the methods of calculating square roots use this fact, but the way in which it is employed is quite different in various calculators and computers.

The program here calculates an upper and lower limit for the square root of a number and, by successive approximations, pinches the root to within a smaller and smaller interval until it reaches the desired level of accuracy.

The starting value for the lower limit is 0 and for the upper limit the number, Z, whose square root is sought. The program then divides this interval into ten steps by simply dividing the difference between the numbers by 10. The variable I is increased from the lower limit to the upper one by the value of the step, S. At any point, if I squared becomes greater than Z, a new upper limit is set to I and a new lower limit is set to I - S.

This method converges very quickly and adds approximately one decimal place of accuracy with each pass beyond the third. What happens when you enter into the program a number that has an exact square root such as 25 or 49? Why?

Another approach to calculating square roots by successive approximations is to start with a trial root, X. If X * X is less than the original number N, then increase the trial value by a 0.1. If X * X is greater than N, return to the previous value. This is the first digit of the root. Now, start advancing by 0.01. Continuing in this way, one digit is developed at a time until the desired precision is reached.

This method is quite suitable and fast for square roots of numbers less than 1. A good first trial root value is 0.1. But is it suitable for larger numbers? How should a starting trial value be determined? In this method, especially for numbers greater than 10, the initial trial value for the root matters a great deal in determining the length of time it will take for the calculation to converge. Write a program using this method and compare the speed and accuracy with the program in the book.

```
   5 CLS
  10 REM A=LOWER LIMIT
  20 REM B=UPPER LIMIT
  30 REM Z=STEP IN INTERVAL
  40 REM E=THE ACCURACY YOU DESI
RE
  50 LET E=.00001
  60 PRINT "PROGRAM FINDS SQUARE
 ROOT OF ANY"
  70 PRINT "POSITIVE NUMBER BY P
INCHING IT"
  80 PRINT "WITHIN A SMALLER AND
 SMALLER"
  90 PRINT "INTERVAL"
 100 PRINT
 110 PRINT "INPUT NUMBER"
 120 INPUT Z
 130 IF Z<=0 THEN GOTO 120
 135 CLS
 140 PRINT "LOWER LIMIT";TAB 20;
"UPPER LIMIT"
 150 PRINT "-----  -----";TAB 20;
"-----  -----"
 160 LET A=0
 170 LET B=Z
 180 LET S=(B-A)/10
 190 PRINT A;TAB 10;"<EXACT RT<"
;TAB 20;B
 200 IF ABS (A*B-Z)<E THEN GOTO
290
 210 FOR I=A TO B STEP S
 220 IF Z<I*I THEN GOTO 260
 230 NEXT I
 240 LET B=B*10
 250 GOTO 180
 260 LET B=I
 270 LET A=I-S
 280 GOTO 180
 290 PRINT
 300 PRINT "APPROXIMATION NOW CO
RRECT TO AN ACCURACY OF ";E
 310 PRINT "YOU MAY USE EITHER "
;A;" OR",B;" AS THE SQUARE","ROO
T OF ";Z
 320 PAUSE 4E4
 330 IF INKEY$="1" THEN RUN
```

```
PROGRAM FINDS SQUARE ROOT OF ANY
POSITIVE NUMBER BY PINCHING IT
WITHIN A SMALLER AND SMALLER
INTERVAL

INPUT NUMBER

54

LOWER LIMIT              UPPER LIMIT
-----  -----             -----  -----
0           <EXACT RT<54
5.4         <EXACT RT<10.8
7.02        <EXACT RT<7.56
7.344       <EXACT RT<7.398
7.344       <EXACT RT<7.3494
7.34832     <EXACT RT<7.34886
7.348428    <EXACT RT<7.348482
7.3484658   <EXACT RT<7.3484712
7.348469    <EXACT RT<7.3484696

APPROXIMATION NOW CORRECT TO AN
ACCURACY OF .00001
YOU MAY USE EITHER 7.348469 OR
7.3484696 AS THE SQUARE
ROOT OF 54



PROGRAM FINDS SQUARE ROOT OF ANY
POSITIVE NUMBER BY PINCHING IT
WITHIN A SMALLER AND SMALLER
INTERVAL

INPUT NUMBER

39

LOWER LIMIT              UPPER LIMIT
-----  -----             -----  -----
0           <EXACT RT<39
3.9         <EXACT RT<7.8
6.24        <EXACT RT<6.63
6.24        <EXACT RT<6.279
6.2439      <EXACT RT<6.2478
6.24468     <EXACT RT<6.24507
6.244992    <EXACT RT<6.245031
6.2449959   <EXACT RT<6.2449998

APPROXIMATION NOW CORRECT TO AN
ACCURACY OF .00001
YOU MAY USE EITHER 6.2449959 OR
6.2449998 AS THE SQUARE
ROOT OF 39
```

5

Compounding

As with successive approximations and recursion, compounding requires many repetitive calculations. Some compound interest and growth situations can be represented by a formula, but for many problems, solving by repetitive calculations is an excellent approach.

There is nothing magical about compounding; you generally start with an initial amount of money, number of animals, etc. This quantity then grows or diminishes by a certain percentage at set intervals. The new amount is the old amount increased or decreased by this percentage. Repeat this calculation over and over again, and you have solved the problem.

Five different types of problems which involve compounding of some sort are described in this chapter. Try the extra problems that follow some of the programs; you may be surprised at the results.

Indians and Interest

Here is a simple compound interest problem that produces an astonishing answer. The problem has to do with the sale of Manhattan Island to the Dutch for approximately $24 worth of trinkets and beads.

If the $24 that the Indians received in 1626 had been deposited in a bank paying 5-¾% interest compounded annually, how much would it amount to in 1983?

The program here solves this little problem by making use of the formula to calculate compound interest. In particular, if P dollars are invested at an interest rate of R (expressed as a decimal) and compounded N times, then the total amount A is given by the formula:

$$A = P(1 + R)^N$$

How much was gained in 1983 alone? How much was gained in the decade from 1974 to 1983? You can change the value of N in Statement 30 to get the answers to these questions. But a better way might be to enter the ending year with an INPUT statement.

There are many other ways you can improve the program and make it more suitable for general purpose compound interest calculations. Modify it to accept any starting and ending year, any rate of interest, and any starting principal amount.

The problem as stated is somewhat unrealistic since banks were not paying interest rates of 5-¾% from 1626 to 1983. Change the program to calculate the total amount based on the following interest rates:

Years	*Interest Rate*
1626-1830	1.5%
1831-1870	2.0%
1871-1910	3.0%
1911-1921	3.5%
1922-1929	6.5%
1930-1940	2.3%
1941-1945	3.5%
1946-1960	5.3%
1961-1980	6.5%
1981-1983	9.5%

```
 10 LET P=24
 20 LET R=.0575
 30 LET N=1983-1626
 40 LET A=P*((1+R)**N)
 50 PRINT "IF THE $24 WHICH THE
INDIANS"
 60 PRINT "RECEIVED IN 1626 FOR
MANHATTAN"
 70 PRINT "ISLAND HAD BEEN DEPO
SITED IN A"
 80 PRINT "BANK PAYING ";R*100;
" PERCENT"
 90 PRINT "INTEREST COMPOUNDED
ANNUALLY, IT"
 100 PRINT "WOULD HAVE AMOUNTED
TO APPROXI-"
 110 PRINT "MATELY $";A;" BY 198
3."
```

```
IF THE $24 WHICH THE INDIANS
RECEIVED IN 1626 FOR MANHATTAN
ISLAND HAD BEEN DEPOSITED IN A
BANK PAYING 5.75 PERCENT
INTEREST COMPOUNDED ANNUALLY, IT
WOULD HAVE AMOUNTED TO APPROXI-
MATELY $11176492000 BY 1983.
```

Systematic Savings

In the previous program, compound interest was calculated by means of a formula well known to bankers, money lenders and real estate agents. However, if you did not know the formula, how would you calculate interest on money in a savings account by hand or with a calculator?

You would probably begin by multiplying the principal amount P by the interest rate R and adding that amount to the original principal at the start of the second year. Doing that for all the years the money is in the bank will yield a final amount. Why not write a program to perform the calculations in this manner rather than use a formula? Here is such a program.

It is set up for an initial principal amount of 100 (Statement 10), an interest rate of 10% (R = 0.1 in Statement 20) and ten years (N = 10 in Statement 30). Naturally these values could be read in using INPUT statements.The clever calculation in Statement 60 rounds off the amount to two decimal places (dollars and cents).

In contrast with the program in the previous section, this one does not use a compound interest formula, but simply adds the interest each year to the growing principal amount in Statement 50.

```
   5 PRINT "AT END OF YEAR","BAL
ANCE"
  10 LET P=100
  20 LET R=.1
  30 LET N=10
  40 FOR I=1 TO N
  50 LET P=P+P*R
  60 PRINT I,INT (P*100+.5)/100
  70 NEXT I

AT END OF YEAR   BALANCE
1                110
2                121
3                133.1
4                146.41
5                161.05
6                177.16
7                194.87
8                214.36
9                235.79
10               259.37
```

Now, how could this program be modified to allow for a plan of systematic saving? In other words, instead of letting the $100 lie around all lonely while it is compounding, each year you add another $100 to it. With this program, making the modification for systematic savings is easy: Statement 80 is added to add the new deposit each year to the ever growing principal.

```
   5 PRINT "END OF YEAR  INVESTE
D   TOTAL"
  10 LET N=10
  20 LET C=100
  30 LET R=.1
  40 LET P=C
  50 FOR I=1 TO N
  60 LET P=P+P*R
  70 PRINT I;TAB 13;I*C;TAB 24;I
NT (P*100+.5)/100
  80 LET P=P+C
  90 NEXT I
```

```
END OF YEAR   INVESTED    TOTAL
1             100         110
2             200         231
3             300         364.1
4             400         510.51
5             500         671.56
6             600         848.72
7             700         1043.59
8             800         1257.95
9             900         1493.74
10            1000        1753.12
```

In an effort to attract depositors, many banks over the last 20 years have started offering savings and investment accounts in which the deposits are compounded more often than annually. In the early 50's, many banks went to quarterly compounding; in the late 50's, to daily; and in the early 60's, some "competitive" S&L's went to continuous compounding.

How much does more frequent compounding really mean? Modify either of the two programs to compute the interest for more frequent compounding. What is the difference in interest for one year for annual, quarterly, monthly, daily, and continuous (every second) compounding on a principal amount of $1000 invested at 8%? How about for ten years?

Incidentally, if you want to use the formula method in the previous section for this calculation, the formula for P principal invested at R rate compounded N times per year is:

$$A = P(1 + R/N)^N$$

Try this problem which uses the same principles of compounding. Consumer prices rose an average of 8.8% in 1980. While the government keeps trying to bring inflation under control, they don't seem to be meeting with much success. Assuming that prices continue to go up this much (8.8%) every year, how much will a $6000 economy car (1980 dollars) cost in the year 2000? How much will it cost when you are 65 years old?

Systematic Savings Revisited

Using the formula for compound interest and systematic savings, this program will calculate the amount accumulated after a given period of time.

When you run the program, it will ask how much you wish to save each month, the number of compounding periods in a year, the interest rate, and the length of time you wish to continue your systematic savings program. The program will then calculate the total amount at the end of that period.

From the preceding programs, you should be able to see that a systematic savings program is a very effective way to accumulate a nest egg for the future.

```
  10 PRINT "THIS PROGRAM WILL CA
LCULATE THE"
  20 PRINT "BALANCE OF A SAVINGS
 ACCOUNT IN"
  30 PRINT "WHICH DEPOSITS ARE M
ADE REGULAR-"
  40 PRINT "LY."
  50 PRINT
  60 PRINT "HOW MUCH WILL YOU SA
VE EACH"
  70 PRINT "MONTH?  ";
  80 INPUT A
  90 PRINT A
 100 PRINT "HOW MANY INTEREST CO
MPOUND","PERIODS EACH YEAR?  ";
 110 INPUT B
 120 PRINT B
 130 PRINT "AT WHAT RATE OF INTE
REST","(XX.X PERCENT)?  ";
 140 INPUT C
 150 PRINT C
 160 PRINT "HOW MANY YEARS WILL
YOU MAKE","THIS SAVING?  ";
 170 INPUT D
 180 PRINT D
 190 LET F=0
 200 LET A=A*(12/B)
 210 LET E=(C/100)/B
 220 LET G=B*D
 230 LET T1=0
 240 LET T1=T1+1
 250 IF T1=G+1 THEN GOTO 280
 260 LET F=(1+E)*(F+A)
 270 GOTO 240
 280 LET F=INT (100*F+.5)/100
 290 PRINT
 300 PRINT
 310 PRINT "AFTER ";D;" YEARS YO
U WILL HAVE"
 320 PRINT "$";F
 330 PAUSE 4E4
 340 CLS
 350 GOTO 50
```

```
HOW MUCH WILL YOU SAVE EACH
MONTH?  20
HOW MANY INTEREST COMPOUND
PERIODS EACH YEAR?  2
AT WHAT RATE OF INTEREST
(XX.X PERCENT)?  5.5
HOW MANY YEARS WILL YOU MAKE
THIS SAVING?  10

AFTER 10 YEARS YOU WILL HAVE
$3230.14


HOW MUCH WILL YOU SAVE EACH
MONTH?  10
HOW MANY INTEREST COMPOUND
PERIODS EACH YEAR?  1
AT WHAT RATE OF INTEREST
(XX.X PERCENT)?  3.75
HOW MANY YEARS WILL YOU MAKE
THIS SAVING?  6

AFTER 6 YEARS YOU WILL HAVE
$820.63


HOW MUCH WILL YOU SAVE EACH
MONTH?  10
HOW MANY INTEREST COMPOUND
PERIODS EACH YEAR?  365
AT WHAT RATE OF INTEREST
(XX.X PERCENT)?  5
HOW MANY YEARS WILL YOU MAKE
THIS SAVING?  6

AFTER 6 YEARS YOU WILL HAVE
$839.71
```

Try to combine the things that you have learned from the programs in these sections to write a completely generalized program for systematic savings. It should accept the following information as input:

- Initial deposit
- Frequency of periodic deposits
- Amount of each deposit
- Interest rate
- Interest compounding frequency
- Length of time of savings program

Your program should produce output in tabular form showing years (or other time periods), amount invested without interest, total amount with interest, and the interest alone.

Loan Payments

Although saving money in a systematic way is a noble goal, many people frequently find themselves talking to a different bank officer, namely the one in charge of loans.

This program will calculate the payments for a loan for a period of one year or longer. The program asks you to enter the key facets of the loan in question: amount borrowed, annual rate of interest, interval between payments and term of the loan in years.

The sample run shows a relatively short-term loan (2 years) of $3000 at a bargain basement interest rate of 8%. The monthly payment of the loan is computed to be $135.68 and the total interest $256.34. Why is the total interest not $240 (8% of $3000)? Instead it is 8.54% Is this a mistake in the program?

Try the program for some longer term loans at real world interest rates. For example, run it for an automobile loan of $8000 at 12% over a 5-year period. Perhaps you can see from this that systematically saving your money and paying cash for an item makes more sense than time payments.

Run the program to calculate the mortgage payments on a $150,000 house with a $40,000 down payment. Try it with a 16% interest rate stretched over a 30-year period. Ouch! Look at all that interest!

At the heart of this program are Statements 270-295. Can you see what is happening in these calculations?

```
  10 PRINT "THIS PROGRAM WILL DE
TERMINE"
  20 PRINT "PAYMENTS FOR A LONG
TERM LOAN."
  30 PRINT
  40 PRINT "INPUT AMOUNT BORROWE
D ($)"
  50 INPUT A
  60 PRINT A
  70 PRINT "INPUT INTEREST CHARG
ED (PERCENT)"
  80 INPUT I
  90 PRINT I
 100 PRINT "INPUT INTERVAL BETWE
EN PAYMENTS (MONTHS)  ";
 110 INPUT P
 120 PRINT P
 130 PRINT "TERM OF THE LOANS (Y
EARS)  ";
 140 INPUT Y
 150 PRINT Y
 160 PRINT
 170 PRINT "WHICH DO YOU WISH TO
 SEE?"
```

```
 180 PRINT "1. TOTALS ONLY"
 190 PRINT "2. TABLE + TOTALS"
 200 INPUT V
 210 IF V=1 THEN GOTO 270
 220 PRINT TAB 8;"PRINC";TAB 24;
"PRINC"
 230 PRINT TAB 8;"AT BEG","INT D
UE REPAID"
 240 PRINT "PERIOD  OF PER","END
 PER END PER"
 250 PRINT "------  ------  ----
--- -------"
 260 PRINT
 270 LET Z=(Y*12)/P
 280 LET K=(I*(P/12))/100
 290 LET E=A*K/(1-1/(1+K)**Z)
 295 LET E=INT  (E*100+.5)/100
 300 LET C=A
 310 LET F=0
 320 LET D1=0
 330 LET T1=0
 340 LET T1=T1+1
 350 IF T1>Z THEN GOTO 470
 360 LET B=T1
 370 LET C=C-F
 380 LET D=C*K
 390 LET F=E-D
 400 LET C=INT  (C*100+.5)/100
 410 LET D=INT  (D*100+.5)/100
 420 LET F=INT  (F*100+.5)/100
 430 LET D1=D1+D
 440 IF V=1 THEN GOTO 340
 445 IF B>3 THEN SCROLL
 450 PRINT B;TAB 8;C;TAB 16;D;TA
B 24;F
 460 GOTO 340
 470 IF V=2 THEN GOTO 530
 480 SCROLL
 490 LET D1=INT  (D1*100+.5)/100
 500 PRINT "TOTAL INTEREST PAID:
 $";D1
 505 SCROLL
 510 PRINT "TOTAL PRINCIPAL REPA
ID:  $";A
 520 GOTO 550
 530 SCROLL
 535 PRINT ,"------- -------"
 540 SCROLL
 545 PRINT "TOTALS";TAB 16;D1;TA
B 24;A
 550 LET E5=INT  ((D1+A)*100+.5)/
100
 560 SCROLL
 570 LET E6=E5/((Y*12)/P)
 580 LET E6=INT  (100*E6+.5)/100
 590 PRINT "MONTHLY PAYMENT IS $
";E6
 600 SCROLL
 610 PRINT "AND TOTALS $";E5
```

```
THIS PROGRAM WILL DETERMINE
PAYMENTS FOR A LONG TERM LOAN.

INPUT AMOUNT BORROWED ($)
3000
INPUT INTEREST CHARGED (PERCENT)
8
INPUT INTERVAL BETWEEN PAYMENTS
(MONTHS)  1
TERM OF THE LOANS (YEARS)  2

WHICH DO YOU WISH TO SEE?
1. TOTALS ONLY
2. TABLE + TOTALS
          PRINC                    PRINC
          AT BEG    INT DUE        REPAID
PERIOD    OF PER    END PER        END PER
------    ------    -------        -------

1         3000      20             115.68
2         2884.32   19.23          116.45
3         2767.87   18.45          117.23
4         2650.64   17.67          118.01
5         2532.63   16.88          118.8
6         2413.83   16.09          119.59
7         2294.24   15.29          120.39
8         2173.85   14.49          121.19
9         2052.66   13.68          122
10        1930.66   12.87          122.81
11        1807.85   12.05          123.63
12        1684.22   11.23          124.45
13        1559.77   10.4           125.28
14        1434.49   9.56           126.12
15        1308.37   8.72           126.96
16        1181.41   7.88           127.8
17        1053.61   7.02           128.66
18        924.95    6.17           129.51
19        795.44    5.3            130.38
20        665.06    4.43           131.25
21        533.81    3.56           132.12
22        401.69    2.68           133
23        268.69    1.79           133.89
24        134.8     0.9            134.78
                    -------        -------
TOTALS                256.34       3000
MONTHLY PAYMENT IS $135.68
AND TOTALS $3256.34



TOTAL INTEREST PAID:  $256.34
TOTAL PRINCIPAL REPAID:  $3000
MONTHLY PAYMENT IS $135.68
AND TOTALS $3256.34
```

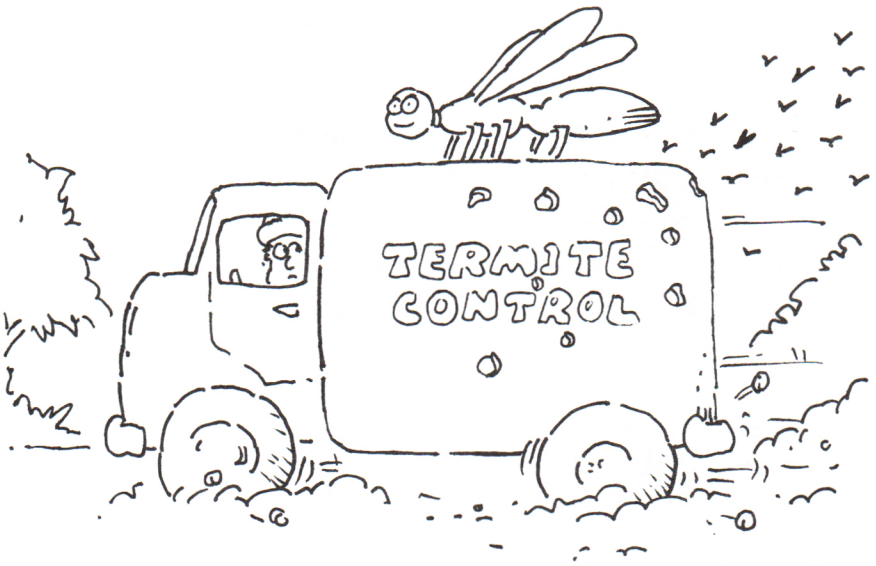
Interest on Credit Purchases

All too frequently, the rate of interest to be charged on a loan or credit purchase is hidden in the small type. After all, the bank or car dealer or finance company wants to convince you that you can afford that new car or home improvement or whatever.

This program calculates the interest rate on a loan given the principal value of the loan, the number of payments, and the amount per payment. To make things even easier, the program will accept the cash purchase price of the article and the down payment and automatically compute the principal value of the loan.

The sample run shows a rather unrealistic loan for an item costing $88.99. The down payment was $10.00 and the loan is over a period of 18 months; each monthly payment is $4.85. The program run shows that the actual interest rate is a modest 5.69%.

Now try the program with a more realistic loan. A termite company in New Jersey advertises a total treatment for your house for only $200; just $29.95 down and 24 monthly payments of $11.95 per month. Is this the bargain that it seems? What is the annual rate of interest?



```
  10 REM THIS PROGRAM DETERMINES
 THE ACTUAL INTEREST YOU PAY
  20 REM WHEN YOU PURCHASE SOMET
HING ON CREDIT
  50 PRINT "INPUT CASH PRICE OF
ARTICLE ($)"
  60 INPUT C
  70 PRINT C
  80 PRINT "INPUT DOWN PAYMENT (
$)  ";
  90 INPUT D
 100 PRINT D
 110 PRINT "INPUT NO. OF PAYMENT
S EXCLUDING DOWN PAYMENT"
 120 INPUT N
 130 PRINT N
 140 PRINT "INPUT NO. OF PAYMENT
S PER MONTH"
 150 INPUT S
 160 PRINT S
 170 PRINT "INPUT AMOUNT PER PAY
MENT ($)"
 180 INPUT R
 190 PRINT R
 200 PAUSE VAL "100"
 210 CLS
 220 LET B=R*N+D
 230 LET I=B-C
 240 LET M=N/(S*12)
 260 LET T=INT (100*(I*100/(B*M)
)+.5)/100
 270 PRINT "RATE OF INTEREST CHA
RGED WAS"
 280 PRINT T;" PERCENT."


INPUT CASH PRICE OF ARTICLE ($)
88.99
INPUT DOWN PAYMENT ($)  10
INPUT NO. OF PAYMENTS EXCLUDING
DOWN PAYMENT
18
INPUT NO. OF PAYMENTS PER MONTH
1
INPUT AMOUNT PER PAYMENT ($)
4.85


RATE OF INTEREST CHARGED WAS
5.69 PERCENT.
```

Population Growth

So far, all the compounding problems in this chapter have involved money—interest, savings, and loans. But many fascinating compounding problems do not involve money. Consider the following problem.

In 1960, the population figures for the United States and Mexico were 180 million and 85 million respectively. The annual population growth rate for the United States was 1.23% and for Mexico 2.23%. If these growth rates remain steady, in some distant year the population of Mexico will exceed that of the United States. In what year will this occur?

The program uses the method of accumulating the principal amount rather than a formula, and applies it to both populations. The sample run reveals that the population of Mexico will overtake that of the U.S. in the not too distant future.

```
 10 LET U=180000000
 20 LET M=85000000
 30 LET R1=1.0123
 40 LET R2=1.0223
 50 LET Y=1960
 60 LET U=U*R1
 70 LET M=M*R2
 80 LET Y=Y+1
 90 IF M<U THEN GOTO 60
100 PRINT "U.S. POPULATION IS "
U,"AND MEXICO IS ";M;" IN ";Y

J.S. POPULATION IS 461406520
ND MEXICO IS 464464690 IN 2037
```

But what if the Mexican people were diligently trying to bring their increasing population under control and their annual growth rate was increasing by only 0.001% per year, compared to the increase of the U.S. growth rate of 0.01% per year. If that were the case, would the population of Mexico ever exceed that of the U.S.?

Insert Statements 60, 70, 85, 110, and 120 into the first program, run it and find out the answer to the question.

```
 60 LET U=U*(1.01*R1)
 70 LET M=M*(1.001*R2)
 85 IF Y>10000 THEN GOTO 120
110 STOP
120 PRINT "U.S. STILL HAS MORE
EOPLE,",U;" THAN MEXICO,",M;" I
N 10000."
```

Going back to the first program, run it with the population data from the U.S. (180 million) and California (15.7 million) from 1960. At that time, the annual growth rates were 1.23% and 3.7% respectively. Running the program will indicate the year that the population of California will exceed that of the United States. This, of course, is nonsense. Where is the discrepancy?

This program might make more sense if it were restated to ask in what year the population of California would exceed that of the remainder of the U.S., if ever?

Compounding can also be used to solve other kinds of population growth problems. Consider the bristleworm. The bristleworm can reproduce by splitting itself into 24 segments, each of which grows a new head and tail. What is the maximum number of bristleworms that could be obtained in this fashion, starting with only one worm, after ten splittings? Assuming a splitting occurs every 22 days, how many offspring will one worm produce in a year? When will the earth be overrun with bristleworms?

Here is another problem for which the principle of compounding is useful. It takes nature about 500 years to produce one inch of topsoil. Many years ago, the United States had an average depth of almost nine inches of this good dirt, but as of 1975, the country was down to about six inches. This type of soil is necessary, of course, for growing food.

Careless management of our soil causes about 1% per year to erode away; it is then lost forever. Once soil depth reaches three inches or less, it is impossible to grow crops on a large scale. Write a program to calculate the year in which the U.S. will have less the 3″ of topsoil, assuming that it continues to erode away at 1% per year.

Will you be alive then? Will your children be alive? Will anyone be alive?

6

Probability

Statistics and probability are subjects that bring up all kinds of images. Some people who have not had a pleasant time in a required college statistics course keep as far away from the subject as possible. To other people, statistics is something with which devious manufacturers can mislead you about their products.

Not long ago, one foreign automobile manufacturer boasted that 90% of their cars sold in the United States in the last seven years were still on the road. This sounds like excellent reliability. But consider the fact that this manufacturer was rapidly expanding in the U.S. market and 65% of the cars they had sold in the U.S. had been sold in just the previous two years. You would expect that all of these would still be running.

If the manufacturer had sold 10% of their 7-year volume in Years 1 to 3, and most of these cars were not running, then the advertising claim loses much of its meaning.

When approached in a logical, step-by-step manner, statistics and probability are not difficult. In fact, they can be a great deal of fun.

Some of the programs use a formula while others simulate the event in which we are interested. The results are the same, but you may find that the simulations help you to understand exactly what is happening.

Pascal's Triangle—Calculated

Pascal's Triangle is quite fascinating. As you can see from the portion of one reproduced below, each row is symmetrical. Each row also contains the coefficients for a binomial expansion. The sums across the ascending diagonals form the Fibonacci sequence. The sums across the rows are all powers of two. Each row corresponds to the digits of a power of 11. Every element is the sum of the two above it. And, in case you care, all the elements in it are identities in combinatorial theory.

The ways this marvelous triangle can be generated are as varied and interesting as its properties, though perhaps more difficult to figure out. Here is one way to use a computer to generate the triangle.

Any element can be found be adding together the two elements immediately above it. The program uses this principle to produce a triangle nine levels deep. Lines 20-40 set the rightmost diagonal to 1. Each element is stored in the two-dimensional variable P(R,C) with R denoting the row and C denoting the "column." The variable T (Line 55) simply leaves some blank spaces so the output resembles a triangle. The crux of the calculation is in Line 70 in which the value of each new element is calculated.

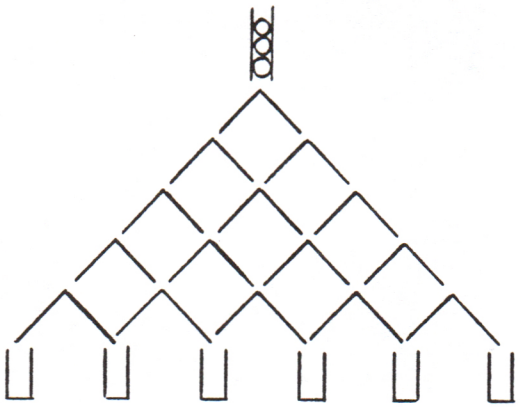
There is another interesting way to calculate Pascal's Triangle in even fewer statements than the program here. It generates the triangle one element at a time and does not use any arrays or two-dimensional variables. Can you figure out how to write such a program?

```
 10 DIM P(11,11)
 20 FOR C=1 TO 10
 30 LET P(C,C)=1
 40 NEXT C
 50 FOR R=1 TO 10
 55 LET T=20-R*2
 60 FOR C=2 TO R+1
 70 LET P(R+1,C)=P(R,C)+P(R,C-1
 75 IF P(R,C)=0 THEN GOTO 90
 80 PRINT TAB T;P(R,C);
 85 LET T=T+3
 90 NEXT C
100 PRINT
105 PRINT
110 NEXT R
```

```
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5   10  10  5   1
        1   6   15  20  15  6   1
      1   7   21  35  35  21  7   1
    1   8   28  56  70  56  28  8   1
```

Pascal's Triangle—by Probability

This program simulates the dropping of balls through a triangular array shown below. At each level, a ball is equally likely to fall either to the left or right. At the bottom of the array a cup is placed at each end point; these cups collect the balls. After each group of balls has been dropped, the number of balls in each cup is tallied and displayed.



```
  5 PRINT "PROGRAM PASCAL"
  6 PRINT
 10 DIM B(10)
 12 CLS
 15 PRINT "HOW MANY BALLS?   ";
 17 INPUT M
 19 PRINT M
 21 PRINT "HOW MANY LEVELS?   ";
 23 INPUT K
 25 PRINT K
 27 PRINT
 30 FOR N=1 TO M
 35 LET T=0
 40 FOR L=1 TO K
 50 IF RND>.5 THEN GOTO 100
 90 LET T=T+1
100 NEXT L
110 LET B(T+1)=B(T+1)+1
120 NEXT N
130 FOR L=1 TO (K+1)
140 PRINT L,B(L)
145 LET B(L)=0
150 NEXT L
160 INPUT I$
170 RUN 6
```

```
HOW MANY BALLS?  1000
HOW MANY LEVELS?  2

1                 250
2                 486
3                 264


HOW MANY BALLS?  1000
HOW MANY LEVELS?  4

1                 75
2                 253
3                 380
4                 232
5                 60
```

The number of balls landing in the various cups at each level, when reduced to the lowest common denominator, should approximate the numbers in Pascal's Triangle. Do you know why they should?

Try a few runs of the program with a different number of balls. Do the numbers obtained really approximate those in Pascal's Triangle? If, at each dividing point in the array, every other ball went in the opposite direction, then Pascal's Triangle would be produced. Try running the program with 4 balls at Level 2; with 8 balls at Level 3; and with 16 balls at Level 4. Do your results look like those of the previous program which calculated the triangle? Does the approximation come closer as the number of balls is increased?

How can you determine how close your results from this program are to the exact value of Pascal's Triangle? One way is to take the number of balls that dropped into each cup on a given level and divide that by the total number of balls divided by the theoretical sum of the row. So, in the sample run at Level 4, you would divide the balls in each cup by 62.5 (1000/16). You then compare that to the "correct" number and compute the percent difference.

Here is the result of this procedure for Level 4 (actually the fifth row) in the sample run:

Cup	*Balls*	*÷62.5*	*Correct*	*Deviation*
1	75	1.20	1	20.00%
2	253	4.05	4	1.25%
3	380	6.08	6	1.33%
4	232	3.71	4	7.20%
5	60	.96	1	4.00%

Why are the outside cups "off" by a greater percentage than those closer to the center? Does this also happen on other levels with different numbers of balls?

Common Birthdays

Here is an interesting little problem in probability. In a group of ten people selected at random, what is the probability that any of them will share the same birthday? How about a group of 20 people? Of 50 people?

Conversely, how many people would you need in a group such that there is a 50% probability that at least two of them have the same birthday? How many people would be needed for a 90% probability of an overlap?

Try to answer these questions, either by guess or by calculation, before you look at the output from the program.

This program provides a painless introduction to the world of statistics. The calculation is actually quite trivial. The probability that any person in a group has a birthday on January 1 is 1/365. If our group has only two people in it, the probability that both of them have a birthday on January 1 is 1/365 times 1/365, or a very small number indeed. The probability that they have a common birthday is 365 times the very small number just obtained, or about 0.27%.

However, if there are three people in the group, the probability goes up slightly. Call the three people Betsy, Ken, and Larry. From the reasoning above, we know that there is a 0.27% probability that Betsy and Ken have the same birthday; also a 0.27 % probability that Betsy and Larry share a birthday; and finally, a 0.27% chance that Larry and Ken have a common birthday. Hence, the total probability for a group of three people is three times the probability for just two people.

A group of four increases the probability over that of two people by a factor of six, five people by a factor of 10, six people by 15, seven people by 21, and so on. There is a progression here, but the probability can also be calculated by the formula:

$$P = 1 - \frac{365\text{-}N}{365}$$

Can you see why this formula produces the same result as the description in words and associated progression above?

Consider all the presidents of the United States. (How many have there been to date?) Two of them, James Polk and Warren Harding, were born on November 2. Is this to be expected given the size of the group?

Since birthdays can be predicted, at least statistically, it ought to be possible to predict deaths as well. It is interesting too that John Adams, James Monroe, and Thomas Jefferson all died on July 4th. Millard Fillmore and William Taft both died on March 8. What is the probability of that set of events?

```
  10 PRINT ,"PROBABILITY THAT"
  20 PRINT "NO. OF","AT LEAST 2
HAVE"
  30 PRINT "PEOPLE","SAME BIRTHD
AY"
  40 PRINT "---------","-------
--------"
  45 LET Z=VAL "365"
  50 LET Q=(Z-1)/Z
  60 FOR N=VAL "2" TO VAL "40"
  70 LET P=VAL "100"*(SGN PI-Q)
  80 PRINT "   ";N,INT (P*VAL "10
0"+.5)/VAL "100";" PERCENT"
  90 LET Q=Q*(Z-N)/Z
 100 LET F=VAL "10"
 110 IF N/F=INT (N/F) THEN PAUSE
 4E4
 120 IF N/F=INT (N/F) THEN CLS
 130 NEXT N
```

```
                    PROBABILITY THAT
NO. OF              AT LEAST 2 HAVE
PEOPLE              SAME BIRTHDAY
---------           ----------------
   2                0.27 PERCENT
   3                0.82 PERCENT
   4                1.64 PERCENT
   5                2.71 PERCENT
   6                4.05 PERCENT
   7                5.62 PERCENT
   8                7.43 PERCENT
   9                9.46 PERCENT
  10                11.69 PERCENT
  35                81.44 PERCENT
  36                83.22 PERCENT
  37                84.87 PERCENT
  38                86.41 PERCENT
  39                87.82 PERCENT
  40                89.12 PERCENT
```

Now it is your turn to write a program. Here is a game to be played among your friends. You make a bet that they have similar preferences in colors. Each person in the group puts up one penny and you, because you are so sure of yourself (and probability), put up an amount in cents equal to the number of people in the group. If you win, you get to keep all of the money; if your friends win, each of them gets double his original bet, or two cents.

Then each person selects a color (from a larger list than there are people, of course). If any two have picked the same color, you win; if all have picked different colors, they win.

Since you would like to win, you need to know how many colors should be on the list for different size groups to give you a better than 50-50 chance of winning. You might like the probability to be up around 70% or so. You can produce the necessary table with a five-line program. Go to it!

Coins in a Pocket

The next two programs were originally written by Glenda Lappan and M.J. Winter and appeared in *Creative Computing* magazine. They are marvelous simulations for illustrating various aspects of probability.

Coins in a Pocket is a simulation of the following situation. A newspaper costs 5 cents. A customer has 5 pennies and a dime in his pocket and offers to pay for his paper by letting you, the vendor, select at random two of the six coins. If you and this customer repeat this procedure for the 20 working days of a month, how much more or less than $1.00 (20 days x 5 cents) are you likely to have collected?

The program below solves this little problem. The first random coin is selected from a group of six (Line 70 in which X = INT(RND*6)). The value of X can be 0, 1, 2, 3, 4, or 5. If it is 0, we assume the dime was selected and a second pick is not made, because it will surely be a penny. Thus 11 cents is added to the running total in Line 130.

If the first coin is a penny (X = 1, 2, 3, 4, or 5) then we make a second pick from the five remaining coins. Again, if the dime is chosen (Y = 0), 11 cents is added to the total; otherwise 2 cents is added.

As you can see from the sample runs, after a great number of trials, the average amount of money collected each day seems to be close to 5 cents. If you would like to convince yourself that the answer is, in fact, 5 cents, consider the following. Assign a letter to each coin, A - F. Let A be the dime and B through F be the pennies. Since all combinations are equally probable, here are all the possible combinations:

AB
AC BC
AD BD CD
AE BE CE DE
AF BF CF DF EF

There are five combinations with a dime (5 x 11 cents) and ten combinations of only pennies (10 x 2 cents). Add it up and you have 55 cents plus 20 cents divided by a total of 15 combinations which equals an average value of 75/15 or 5 cents.

```
   5 PRINT "TAKING TWO COINS AT
RANDOM FROM A DIME AND 5 PENNIES
"
  10 PRINT
  20 PRINT "HOW MANY TIMES SHALL
 WE REPEAT?"
  30 INPUT N
  40 PRINT N
  50 LET U=0
  60 FOR K=1 TO N
  70 LET X=INT (RND*6)
  80 IF X=0 THEN GOTO 130
  90 LET Y=INT (RND*6)
 100 IF Y=0 THEN GOTO 130
 110 LET U=U+2
 120 GOTO 140
 130 LET U=U+11
 140 NEXT K
 150 PRINT
 160 PRINT "AVERAGE VALUE OF THE
 COINS IS"
 170 PRINT U/N;" CENTS"
```

```
TAKING TWO COINS AT RANDOM FROM
A DIME AND 5 PENNIES

HOW MANY TIMES SHALL WE REPEAT?
100

AVERAGE VALUE OF THE COINS IS
4.79 CENTS
```

```
TAKING TWO COINS AT RANDOM FROM
A DIME AND 5 PENNIES

HOW MANY TIMES SHALL WE REPEAT?
1000

AVERAGE VALUE OF THE COINS IS
4.826 CENTS
```

```
TAKING TWO COINS AT RANDOM FROM
A DIME AND 5 PENNIES

HOW MANY TIMES SHALL WE REPEAT?
500

AVERAGE VALUE OF THE COINS IS
5.114 CENTS
```

Baseball Cards

In statistics and probability, it is frequently necessary to predict the number of trials on average until one is successful. For example, if you are trying to roll a four with one die, on average how many tries will it take until you do so?

Since a die has six sides, the probability of rolling any number between 1 and 6 is p = $\frac{1}{6}$. Thus, on any given roll, you have a $\frac{1}{6}$ chance of rolling a four and $\frac{5}{6}$ chance of rolling something else. This failure is designated q. So to be successful in rolling a four, we have a $\frac{1}{6}$ chance on the first roll. On two rolls, the probability is two trials x the probability of one failure, then success ($2 \times \frac{5}{6} \times \frac{1}{6}$). On the third roll, the probability of success is three trials x the probability of two failures followed by one success ($3 \times \frac{5}{6} \times \frac{5}{6} \times \frac{1}{6}$).

Continuing this reasoning leads to the formula for the expected number of trials to success:

$$E = 1p + 2q \times p + 3q \times p + 4q \times p + \ldots$$

This series can be reduced and solved for E which leads to:

$$E = \frac{1}{1\text{-}q} = \frac{1}{p}$$

So now the answer to the original problem can be calculated. The expected number of trials to roll a four is 1/p = 6 tries until success.

But there is another way to approach this type of problem with the assistance of the computer. Consider the problem. Assume there are ten different prizes in Crunchies cereal boxes. How many boxes of cereal would you have to buy to obtain the complete set? The formula above can be expanded to solve this problem:

$$\frac{N}{N} + \frac{N}{N\text{-}1} + \frac{N}{N\text{-}2} + \cdots + \frac{N}{1} = N(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N}$$

For a value of ten, you can solve this formula by hand. However, let's say you would like to solve this problem for baseball cards as found in packs of bubble gum. If there are 50 cards in a complete set, how many packs of gum must you buy, on average, to get a complete set. Write a computer program using the general formula above to solve for a set of any size. The answer for 50 cards is 224.96 packs of gum; how many would you have to buy for a set of 100 cards?

There is another way of approaching this problem. This method is similar to that used to randomly select coins out of the pocket. In this case, the random group is set equal to the total number of cards; this value is accepted as input in Statement 120. Each purchase is set equal to a random number between 1 and N (Statement 210). This card is then put into its proper place in the collection (Statement 220). However, if there is already a card there from

a previous purchase, we simply increment the purchase counter (Statement 250) but do not get any closer to obtaining a complete set. After each purchase, we test to see if the set is complete (Statement 260), otherwise we go on buying more packs of bubble gum.

When the set is complete, the number of packs of gum are tallied up and printed out. After a set of trials, the average is computed. This averaging value should be reasonably close to the value obtained by the formula although it will take a great number of trials before the two numbers are within 1% of each other.

```
 100 DIM C(100)
 110 PRINT AT 15,0;"BASEBALL CAR
DS. THERE ARE N"
 115 PRINT "CARDS IN THE SERIES.
ENTER N."
 120 INPUT N
 125 PRINT N
 130 PRINT "FROM HOW MANY TRIALS
DO YOU WANT"
 135 PRINT "TO COMPUTE THE AVERA
GE? ";
 140 INPUT K
 145 PRINT K
 150 LET S=0
 160 FOR I=1 TO K
 170 LET D=0
 180 FOR J=1 TO N
 190 LET C(J)=0
 200 NEXT J
 210 LET X=INT (RND*N)+1
 220 LET C(X)=C(X)+1
 230 IF C(X)=1 THEN GOTO 250
 240 GOTO 210
 250 LET D=D+1
 260 IF D=N THEN GOTO 280
 270 GOTO 210
 280 LET T=0
 290 FOR J=1 TO N
 300 LET T=T+C(J)
 310 NEXT J
 320 SCROLL
 325 PRINT T;" CARDS"
 330 LET S=S+T
 340 NEXT I
 342 SCROLL
 345 SCROLL
 350 PRINT "AVERAGE NUMBER OF CA
RDS IS"
 355 SCROLL
 360 PRINT S/K
```

```
BASEBALL CARDS. THERE ARE N
CARDS IN THE SERIES. ENTER N.
10
FROM HOW MANY TRIALS DO YOU WANT
TO COMPUTE THE AVERAGE?  15
35 CARDS
28 CARDS
19 CARDS
24 CARDS
20 CARDS
26 CARDS
20 CARDS
21 CARDS
21 CARDS
21 CARDS
23 CARDS
25 CARDS
39 CARDS
22 CARDS
41 CARDS
AVERAGE NUMBER OF CARDS IS
25.666667
```

If you run this second program for a set of 100 cards, it will sometimes run for a very long time before arriving at the answer. Try it with a set of 500 cards as are used by some real bubble gum manufacturers. You can go have your dinner while the program computes just the first value. Be sure to change the dimension statement in Line 100 to C(500).

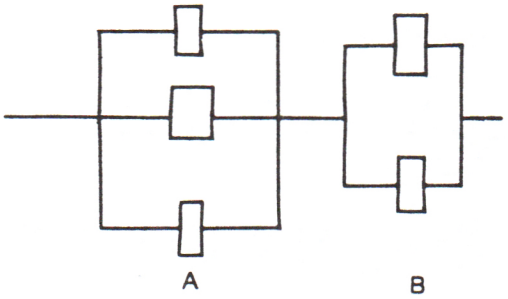
System Reliability

As more and more people in the world come to depend upon mechanical and electronic devices in a myriad of different ways, it is important that these technological devices continue to function.

Some years ago, the military came up with a measure that could be applied to all kinds of systems, big and small, to measure reliability. It is called "mean time between failures." What this means is the length of time, on average, between breakdowns. For a tank, this may be 100 hours, while for a spacecraft, the MTBF must be considerably longer than the planned mission.

As we saw in an earlier section, it is frequently desirable to break down a large problem into smaller subproblems. To calculate the MTBF for a spacecraft would be quite impossible. Instead, it is necessary to start with smaller systems and build up to the whole.

Consider one of the electrical subsystems of a spacecraft. It uses five components as shown below, two parallel systems A and B, arranged in series. The parallel subsystems are said to be "redundant." This is one method of increasing reliability since the system will continue to work if at least one of the components works.



If the manufacturer of the components stated that each one has a 60% probability of lasting 1000 hours, what is the chance that the entire system will last 1000 hours? Take a guess before reading on to the solution below. Is your guess greater than 60% or less? Why?

The program below is a simulation of this system. Remember in subsystem A, it will continue to work if any of the three components works and in B if either of the two components works. However, the system will fail if all three of the A components or both of the B components fail.

In Statement 10, the program is set up to make 500 trials of the system. Statement 30 selects a random integer between 1 and 10 for each component for each trial. If the value of this integer is 6 or under, the component is okay (60% probability of working at the end of 100 hours). If it is 7, 8, 9, or 10, this means the component has failed.

```
  1 DIM C(5)
  5 LET S=0
 10 FOR I=1 TO 500
 20 FOR J=1 TO 5
 30 LET C(J)=INT (RND*10)+1
 40 IF C(J)>6 THEN GOTO 70
 50 LET C(J)=1
 60 GOTO 80
 70 LET C(J)=0
 80 NEXT J
 90 IF C(1)=1 OR C(2)=1 OR C(3)
=1 THEN GOTO 110
 100 GOTO 140
 110 IF C(4)=1 OR C(5)=1 THEN GO
TO 130
 120 GOTO 140
 130 LET S=S+1
 140 NEXT I
 150 PRINT "RELIABILITY ABOUT ";
S/500

RELIABILITY ABOUT 0.81


RELIABILITY ABOUT 0.764
```

Statement 90 tests whether subsystem A works and Statement 110 checks out subsystem B. If both work, a success is counted in Statement 130.

There are, of course, shorter ways to do this. For example, all the individual tests (Statements 90 to 120) can be replaced by one overall test (new Statement 90).

```
  90 IF (C(1)+C(2)+C(3))*(C(4)+C
(5))=0 THEN GOTO 110
 100 LET S=S+1
 110 NEXT I

RELIABILITY ABOUT 0.762


RELIABILITY ABOUT 0.788
```

Now it is your turn to use what you have learned in this chapter and write a program to determine the mean time between failures of the system above.

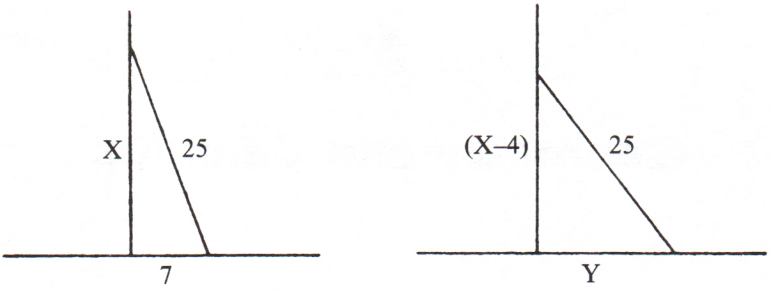
7

Geometry and Calculus

In this chapter several of the problem solving approaches from earlier chapters are used to solve geometric problems. You will find that many problems can be solved in a variety of different ways—by applying a formula, by trial and error, by successive approximations, and, in some cases, by common sense. Perhaps the most important thing to learn from these problems and programs is how to analyze a problem to reach the solution quickly and painlessly.

Crossed and Slipping Ladders

Here is a simple problem of a slipping ladder. A ladder 25 feet long is placed so its foot is 7 feet from the base of a building. The base of the ladder slipped on some loose gravel so that the top is 4 feet lower than where it was to start. How far did the foot of the ladder slip?



The diagrams show the two positions of the ladder. By the Pythagorean theorem we know that $a^2 + b^2 = c^2$, hence, the equations needed to solve the problem are:

$$x = \sqrt{25^2 - 7^2}$$

$$y = \sqrt{25^2 - (x\text{-}4)^2}$$

and the amount of slippage of the base is $z = y - 7$. These three equations are put into a computer program which quickly calculates an answer of 8 feet.

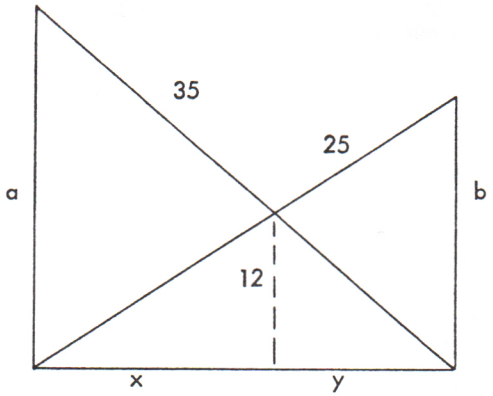
While the aritmetic in the above problem is not particularly messy, it is still no great joy to solve by hand. However, the computer is just as happy to do the problem with really messy dimensions, say a ladder length of 27.83 feet and a distance from the wall of 7.62 feet.

```
    5 PRINT "PROGRAM LADDER"
    6 PRINT
   10 LET X=SQR (25*25-7*7)
   20 LET Y=SQR (25*25-(X-4)*(X-4
))
   30 LET Z=Y-7
   40 PRINT "LADDER BASE SLIPPED
";Z;" FEET."
```

```
PROGRAM LADDER

LADDER BASE SLIPPED 8 FEET.
```

```
10 LET X=SQR (27.63**2-7.62**2
)
20 LET Y=SQR (27.63**2-(X-4)**
2)
30 LET Z=Y-7.62

PROGRAM LADDER

LADDER BASE SLIPPED 8.3861316
FEET.
```

Let's consider an old problem found in many classic collections. Two ladders, one 25 feet long and the other 35 feet long lean against buildings on opposite sides of an alley as shown below. The point at which the ladders cross is 12 feet above the ground. How wide is the alley?



By using similar triangles twice we find that

$$12/a + 12/b = 1$$

Then, by applying the Pythagorean theorem and reducing, we obtain:

$$a^2 - b^2 = 600$$

Using one of the methods described in the Problem Solving chapter, you can solve these two simultaneous equations. Or they can be combined into one equation in which z, the width of the alley is

$$z = \sqrt{35^2 - a^2}$$

By eliminating b, the following fouth degree equation is obtained

$$a^4 - 24a^3 - 600a^2 + 14400a - 86400 = 0$$

Again, this can be solved using one of the methods in the chapter on Problem Solving. This is the traditional approach, but there is another. The solution is the intersection of the curves described by the two original equations. By using successive approximations, say in steps of 1 for a, you could solve for b_1 and b_2 in the following restated original equations

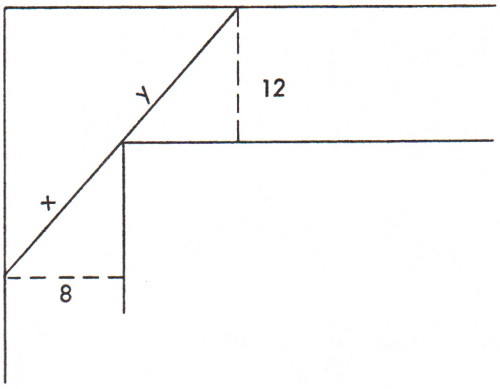
$$b_1 = \frac{12a}{a - 12}$$

$$b_2 = \sqrt{600 - a^2}$$

When b_1 falls below b_2, reduce the step to 0.1 to obtain a closer approximation. By continuing this method of successive approximations, it is possible to obtain a very accurate solution.

Is there another approach? Yes, there is and it also avoids the quartic equation. It uses the original equations in a trial and error procedure as described in the chapter on Sets and Repetitive Trials. See if you can write a program using this approach.

Here is another classic problem for you to solve in any way that you like. What is the longest ladder that can be carried in a horizontal position around the corner made where a 12-foot wide alley meets one that is 8 feet wide. The diagram shows the problem.



Distance Between Coordinate Points

This program solves for the distance between any two points in three-dimensional space defined by their x, y, and z coordinates.

The formula for the distance between two points in three-dimensional space is:

$$d = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$$

It would be desirable for the program to be able to solve any problem of this kind. One approach would be to use INPUT statements to accept the point coordinates. Another is to use a DATA statement to define the points. Then only this one statement has to be changed for a new problem or set of problems.

Since the Timex 1000 does not have DATA statements, we have used the alternative technique described earlier in the book. The program is written to calculate the distance between three sets of points. The points used were:

0, 0, 0	and	3, 4, 5
3.5, -4.7, 6.2	and	-0.9, 3.0, 4.4
67, 36, 82	and	54, 25, 90

This calculation is used extensively in aerospace navigation. Can you determine the angle or "compass heading" of the resulting flight path? Better start with just two dimensions.

```
10 DIM A(18)
15 FOR N=1 TO 18
20 INPUT A(N)
30 NEXT N

INPUT THE FOLLOWING NUMBERS:

0,0,0,3,4,5,3.5,-4.7,6.2,-0.9,3.
4.4,67,36,82,54,25,90
```

```
10 FOR N=1 TO 3
15 PRINT
20 FOR I=6*N-5 TO 6*N
25 LET A=A(I)
27 POKE 16578,PEEK 16578+1
30 NEXT I
35 POKE 16578,CODE "A"
45 LET D1=SQR ((A-D)*(A-D)+(B-
E)*(B-E)+(C-F)*(C-F))
50 PRINT "THE DISTANCE BETWEEN
","("A","B","C") AND ("D,
","E","F")","IS "D1
60 NEXT N
```

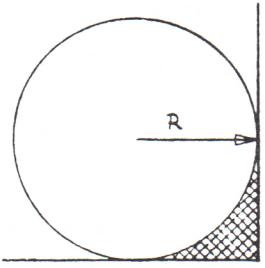
```
THE DISTANCE BETWEEN
(0,0,0) AND (3,4,5)
IS 7.0710678

THE DISTANCE BETWEEN
(3.5,-4.7,6.2) AND (-0.9,3,4.4)
IS 9.0493094

THE DISTANCE BETWEEN
(67,36,82) AND (54,25,90)
IS 18.814858
```

Note: Use same instructions as for the exponential equation solver on page 33.

Area—by Calculation

It is a simple matter to calculate the area of regular geometric figures by using the usual formulae. However, the problem becomes more difficult when it is necessary to calculate the area of two combined regular shapes or the area of irregular shapes. This program shows the method of analysis for solving a problem of the first type, while the next program demonstrates four methods of dealing with irregular areas.

The problem is to solve for the shaded area of the figure below for any value of the radius, R.



The first step is to recall the formulae for calculating the area of a circle and square:

$$A \text{ (circle)} = \text{pi} * R^2$$

$$A \text{ (square)} = \text{Side}^2 \text{ or } (2 * R)^2$$

The difference in area between a square and a circle inscribed within its borders is:

$$A \text{ (difference)} = A \text{ (square)} - A \text{ (circle)}$$

and the area of one corner is the difference divided by 4. The program below will calculate the area for any radius.

Now it is your turn. Write a program to calculate the difference in area between a circle and square in which the square is inscribed within the circle. Now, change your program to calculate the difference in area for a triangle, a hexagon and an octagon inscribed within a circle.

```
  5 PRINT "PROGRAM AREA-1"
  6 PRINT
  7 PRINT "WHAT IS THE RADIUS?
";
  8 INPUT R
  9 PRINT R
 10 PRINT
 20 LET A1=PI*R**2
 30 LET A2=(2*R)**2
 40 LET D=(A2-A1)/4
 50 PRINT "AREA OF ENCLOSED COR
NER IS",D
```

```
PROGRAM AREA-1

WHAT IS THE RADIUS?  1

AREA OF ENCLOSED CORNER IS
0.21460184

PROGRAM AREA-1

WHAT IS THE RADIUS?  10

AREA OF ENCLOSED CORNER IS
21.460184

PROGRAM AREA-1

WHAT IS THE RADIUS?  2.1586553

AREA OF ENCLOSED CORNER IS
1
```

Extend your program to calculate the difference in area for any regular figure inscribed within a circle of radius 1.0. Set up a table or results as follows:

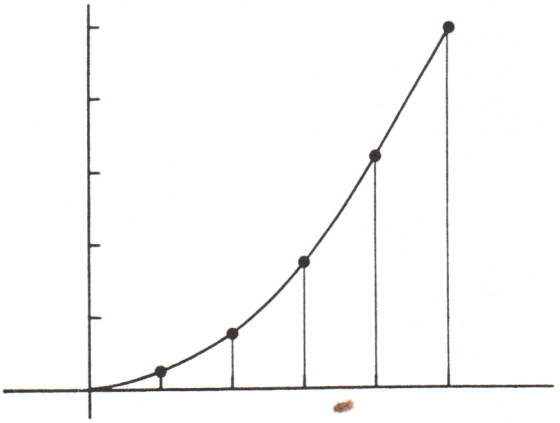
Number of sides	*Difference in Area*
3	
4	
5	
6	
7	
8	

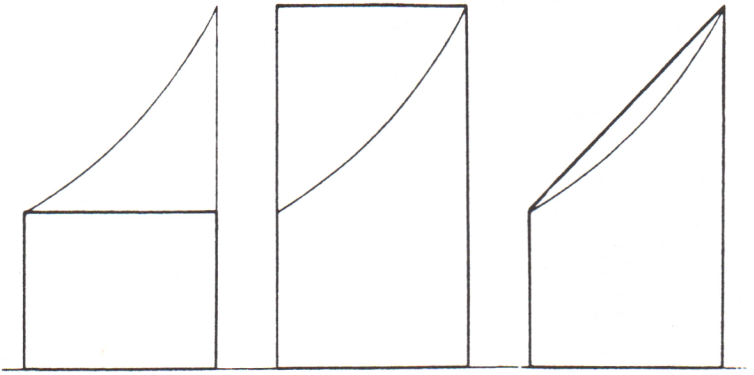
What can you conclude from these results? Do these areas follow in some sort of progression?

Area—by Integration

In many cases it is necessary to determine the area of an irregular figure or the area under a curve where an exact formula is not available. The approach most commonly used is to divide up the enclosed area into small regularly shaped pieces and sum up the areas of all of these pieces.

The easiest shape to use in these calculations is a rectangle. A group of rectangles can either be inscribed within the irregular figure or curve, or circumscribed around it. The first two diagrams show these two methods being used to find the area under a curve. A third method is to use trapezoids which, depending upon the direction of curvature, will either be inscribed or circumscribed automatically.





Inscribed Rectangle

Circumscribed Rectangle

Trapezoid

A fourth method, known as Simpson's Rule, essentially fits a series of parabolas between the points of the curve and calculates the average area. It requires that the area be divided into an even number of parallel slits. Let us call the total number of divisions 2m which are h distance apart. The point on the curve where the first line intersects is y_0, the second y_1, and so on until y_{2m}. The area is then given by the formula:

$$A = \tfrac{1}{3}h[(y_0 + y_{2m}) + 4(y_1 + y_3 + \ldots + y_{2m-1}) + 2(y_2 + y_4 + \ldots y_{2m-2})]$$

The area as calculated by Simpson's Rule converges extremely quickly compared to the other methods. Nevertheless, as the number of intervals increases, all the methods approach the same limit. As might be expected, the trapezoidal approach converges more quickly than either of the approaches using rectangles. Compute the average of the two methods using rectangles. What do you get? Does this suggest another method?

The methods used in this program involve the calculus. And you thought the calculus was difficult! Now you know otherwise.

```
  10 PRINT "WHAT ARE YOUR VALUES
FOR A AND B(SMALLER FIRST:A,B)?
   ";
  20 INPUT A
  30 INPUT B
  40 IF A<=B THEN GOTO 70
  50 PRINT "A MUST BE LESS THAN
B"
  60 GOTO 20
  70 PRINT A;"    ";B
 115 PRINT
 120 PRINT "NO OF 1.SUM  2.SUM
3.SUM  4.SUM"
 130 PRINT "SUBIN   OF      OF
  OF      OF"
 140 PRINT "TERVS RECTS  RECTS
TRAPS  PARAB"
 150 PRINT "----- -----  -----
-----  -----"
 155 PRINT
 160 LET M=-2
 170 LET S=0
 180 LET A$="X*X*X"
 190 LET M=M+3
 200 FOR N=M TO M+2
 210 LET C=5
 220 LET Q=0
 230 LET P=0
 240 LET D=2**N
 250 PRINT D;AT N*2+5,4;
 260 LET H=(B-A)/D
 270 FOR I=0 TO D-1
 280 LET X=A+I*H
```

```
 290 LET P=P+H*VAL A$
 300 LET X=A+I*H+H
 310 LET Q=Q+H*VAL A$
 320 NEXT I
 330 PRINT P;AT 2*N+5,11;Q;AT 2*
N+6,18;
 340 LET T=(P+Q)/2
 350 PRINT T;AT 2*N+5,23;
 360 LET X=A
 370 LET Y=VAL A$
 380 LET X=B
 390 LET Z=VAL A$
 400 LET U=Y+Z
 410 FOR J=2 TO (D-2) STEP 2
 420 LET X=A+J*H
 430 LET U=U+2*VAL A$
 440 NEXT J
 450 LET V=0
 460 FOR K=1 TO (D-1) STEP 2
 470 LET X=A+K*H
 480 LET V=V+4*VAL A$
 490 NEXT K
 500 LET S=(U+V)*(H/3)
 510 PRINT S;AT 2*N+8,0;
 520 NEXT N
 530 IF D<64 THEN GOTO 190
 540 IF ABS ((C-S)/((C+S)/2))>.0
001 THEN GOTO 190
 580 PRINT "WOULD YOU LIKE TO TR
Y DIFFERENT VALUES FOR A AND B (
1-YES, 0-NO)"
 590 INPUT Q5
 600 IF Q5=0 THEN STOP
 610 PAUSE 150
 620 CLS
 630 RUN
```

```
WHAT ARE YOUR VALUES FOR A AND B
(SMALLER FIRST:A,B)?    1    10

NO OF 1.SUM   2.SUM   3.SUM   4.SUM
SUBIN   OF      OF      OF      OF
TERVS RECTS   RECTS   TRAPS   PARAB
----- -----   -----   -----   -----
              5248.6875    2499.75
2    753.1875        3000.9375
              3748.9219    2499.75
4    1501.1719       2625.0469
              3093.0117    2499.75
8    1969.1367       2531.0742
              2788.5498    2499.75
16   2226.6123       2507.5811
              2642.1921    2499.75
32   2361.2234       2501.7078
              2570.4816    2499.75
64   2429.9973       2500.2394

WOULD YOU LIKE TO TRY DIFFERENT
VALUES FOR A AND B (1-YES, 0-NO)
```

8

Science

Using techniques and approaches presented in the previous chapters, this chapter contains five programs in the area of science. One uses a formula to solve simple gas problems, two are drill exercises on the gas laws of Boyle and Charles, and the last two are simulations. You will see that the simulations draw upon many previous techniques such as progressions and repetitive calculations.

Gas Volume

Here is a simple program to produce a not-so-simple table of values for gas volumes.

The volume of a gas varies directly with the absolute temperature T (Kelvin) and inversely with the pressure P. If a certain quantity of gas occupies 500 cubic feet at a pressure of 53 pounds per square foot and an absolute temperature of 500 degrees, what volume will it occupy at 600 degrees absolute temperature and pressures from 100 to 1000 pounds per square foot in increments of 50 pounds?

The original conditions are used to solve for the constant K in Statement 10 (K = V*P/T). Then new volumes are computed for varying pressures with T equal to 600 degrees. The formula used is V = K*T/P.

In the second part of the program, Lines 30 to 80 are replaced to produce a plot of the gas volume for the various pressures.

How would you modify the program to deal with a more general case (i.e., other gasses and different temperatures)? Second, can you write a program that produces a table of values for a gas at different pressures and temperatures?

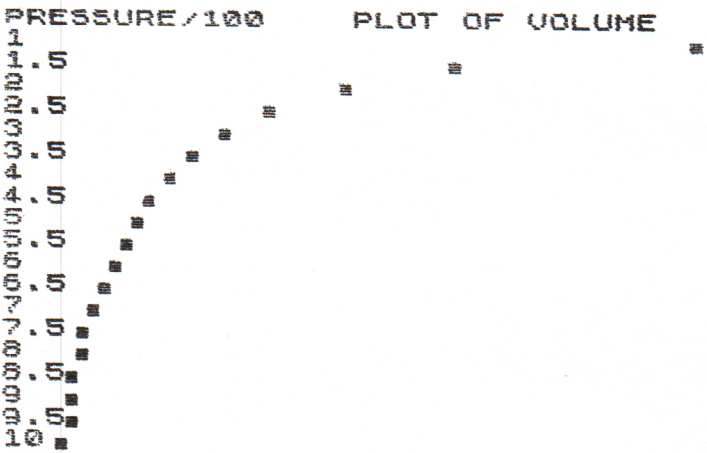
```
  10 LET K=500*53/500
  20 PRINT "A GAS, AT 600 DEGREE
S KELVIN,    AND PRESSURE AS LIST
ED, OCCUPIESTHE VOLUME SPECIFIED
."
  30 PRINT
  40 PRINT "PRESSURE","VOLUME"
  50 FOR P=100 TO 1000 STEP 50
  60 LET V=K*600/P
  65 IF P>900 THEN SCROLL
  70 PRINT P,V
  80 NEXT P
```

A GAS, AT 600 DEGREES KELVIN,
AND PRESSURE AS LISTED, OCCUPIES
THE VOLUME SPECIFIED.

PRESSURE	VOLUME
100	318
150	212
200	159
250	127.2
300	106
350	90.857143
400	79.5
450	70.666667
500	63.6
550	57.818182
600	53
650	48.923077
700	45.428571
750	42.4
800	39.75
850	37.411765
900	35.333333
950	33.473684
1000	31.8

```
  30 PAUSE 4E4
  35 CLS
  40 PRINT "PRESSURE/100","PLOT
OF VOLUME"
  70 PRINT TAB 0;P/100;
  80 LET X=INT (V*2/10+.5)
 100 PLOT X-1,44-P/25
 130 NEXT P
```

A GAS, AT 600 DEGREES KELVIN,
AND PRESSURE AS LISTED, OCCUPIES
THE VOLUME SPECIFIED.



Charles' Law Drill

In the previous program, gas volumes were calculated using Charles' Law and Boyle's Law. Here is a drill and practice program (remember Chapter 1?) that produces problems relating the volume and temperature of a gas. When pressure is constant, the volume/temperature relationships can be stated as follows:

$$\frac{V_0}{T_0} = \frac{V_1}{T_1}$$

After one demonstration problem, the program presents four problems, one to solve for each of the four variables in the equation above.

How can you make the program more efficient? More interesting?

```
  1 CLS
 10 REM : CHARLES
 20 RAND
 90 PRINT TAB 10;"CHARLES LAW"
100 PRINT
110 LET C=0
120 PRINT "INPUT YOUR VALUE FOR
V,V1,T,T1   IN THAT ORDER."
130 PRINT "FOR UNKNOWN VALUES U
SE  0  (ZERO)."
140 PRINT "V   ";
143 INPUT V
146 PRINT V
150 PRINT "V1 ";
153 INPUT V1
156 PRINT V1
160 PRINT "T   ";
163 INPUT T
166 PRINT T
170 PRINT "T1 ";
173 INPUT T1
176 PRINT T1
178 PRINT
180 IF V<0 THEN GOTO 320
185 IF V1<0 THEN GOTO 320
190 IF T<0 THEN GOTO 320
195 IF T1<0 THEN GOTO 320
200 IF V=0 THEN GOTO 240
205 IF V1=0 THEN GOTO 260
210 IF T=0 THEN GOTO 280
215 IF T1=0 THEN GOTO 300
230 PRINT "NO VARIABLES?"
235 PAUSE 200
237 RUN
240 LET V=V1*T/T1
245 PRINT "ORIGINAL VOLUME=";V
250 GOTO 350
260 LET V1=V*T1/T
265 PRINT "NEW VOLUME=";V1
270 GOTO 350
```

```
280 LET T=V*T1/V1
285 PRINT "ORIGINAL TEMPERATURE
=";T
290 GOTO 350
300 LET T1=T*V1/V
305 PRINT "NEW TEMPERATURE=";T1
310 GOTO 350
320 PRINT "NO NEGATIVES."
325 PAUSE 200
330 RUN
350 PAUSE 4E4
352 CLS
355 LET V=INT (RND*50+50)*100
360 LET V1=INT (RND*50+50)*100
370 LET T=INT (RND*125+125)
380 LET T1=INT (RND*125+125)
385 LET C=C+1
390 IF C>=5 THEN GOTO 1000
400 GOTO 390+C*20
410 LET V=0
420 GOTO 500
430 LET V1=0
440 GOTO 540
450 LET T=0
460 GOTO 540
470 LET T1=0
480 GOTO 540
500 PRINT
505 PRINT "VALUES FOR VOLUMES A
RE IN MILLI-LITRES"
510 PRINT
515 PRINT "VALUES FOR TEMPERATU
RES ARE IN KELVIN"
520 PRINT
525 PRINT
530 PRINT "TRY THESE VALUES FOR
V,V1,T,T1:"
540 PRINT "V:";V;TAB 8;"V1:";V1
;TAB 16;"T:";T;TAB 24;"T1:";T1
550 PRINT "WHAT IS YOUR APPROXI
MATE ANSWER USING CHARLES LAW"
560 INPUT Q
565 PRINT Q
570 PRINT "THE CORRECT ANSWER I
S---"
580 GOTO 180
```

```
                CHARLES LAW

INPUT YOUR VALUE FOR V,V1,T,T1
IN THAT ORDER.
FOR UNKNOWN VALUES USE 0 (ZERO)
V   100
V1 0
T   100
T1 300

NEW VOLUME=300



VALUES FOR VOLUMES ARE IN MILLI-
LITRES

VALUES FOR TEMPERATURES ARE IN
KELVIN


TRY THESE VALUES FOR V,V1,T,T1:
V:0      V1:5100 T:238    T1:229
WHAT IS YOUR APPROXIMATE ANSWER
USING CHARLES LAW
5800
THE CORRECT ANSWER IS---
ORIGINAL VOLUME=5300.4367



V:8100  V1:0     T:153    T1:234
WHAT IS YOUR APPROXIMATE ANSWER
USING CHARLES LAW
9300
THE CORRECT ANSWER IS---
NEW VOLUME=12388.235



V:7700  V1:8600 T:0      T1:165
WHAT IS YOUR APPROXIMATE ANSWER
USING CHARLES LAW
150
THE CORRECT ANSWER IS---
ORIGINAL TEMPERATURE=147.73256



V:9800  V1:6800 T:138    T1:0
WHAT IS YOUR APPROXIMATE ANSWER
USING CHARLES LAW
98
THE CORRECT ANSWER IS---
NEW TEMPERATURE=95.755102
```

Boyle's Law Drill

Boyle's Law describes the behavior of gases under ideal conditions when pressure and volume are varied. When the temperature is constant, Boyle found that:

$$P_0 * V_0 = P_1 * V_1$$

Pressure is normally measured in centimeters of mercury while volume is measured in millilitres. As with the drill on Charles' Law, this program presents one sample problem followed by four problems, one to solve for each of the four variables in the equation above.

Unlike many other drill and practice programs, these two do not compare your answer with the correct one. Instead, they leave that up to you to do. Is this desirable? Why or why not? If you feel it is undesirable, change the programs so they do compare answers and calculate a score.

```
  1 CLS
 10 REM : BOYLE
 20 RAND
 90 PRINT TAB 10;"BOYLES LAW"
100 PRINT
110 LET C=0
120 PRINT "INPUT YOUR VALUE FOR
V,V1,P,P1   IN THAT ORDER."
130 PRINT "FOR UNKNOWN VALUES U
SE  0  (ZERO)."
140 PRINT "V   ";
143 INPUT V
146 PRINT V
150 PRINT "V1 ";
153 INPUT V1
156 PRINT V1
160 PRINT "P   ";
163 INPUT P
166 PRINT P
170 PRINT "P1 ";
173 INPUT P1
176 PRINT P1
178 PRINT
180 IF V<0 THEN GOTO 320
185 IF V1<0 THEN GOTO 320
190 IF P<0 THEN GOTO 320
195 IF P1<0 THEN GOTO 320
200 IF V=0 THEN GOTO 240
205 IF V1=0 THEN GOTO 260
210 IF P=0 THEN GOTO 280
215 IF P1=0 THEN GOTO 300
230 PRINT "NO VARIABLES?"
235 PAUSE 200
237 RUN
240 LET V=P1*V1/P
245 PRINT "ORIGINAL VOLUME=";V
250 GOTO 350
```

```
260 LET V1=P*V/P1
265 PRINT "NEW VOLUME=";V1
270 GOTO 350
280 LET P=P1*V1/V
285 PRINT "ORIGINAL PRESSURE=";
P
290 GOTO 350
300 LET P1=P*V/V1
305 PRINT "NEW PRESSURE=";P1
310 GOTO 350
320 PRINT "NO NEGATIVES."
325 PAUSE 200
330 RUN
350 PAUSE 4E4
352 CLS
355 LET V=INT (RND*50+50)*100
360 LET V1=INT (RND*50+50)*100
370 LET P=INT (RND*150+150)
380 LET P1=INT (RND*150+150)
385 LET C=C+1
390 IF C>=5 THEN GOTO 1000
400 GOTO 390+C*20
410 LET V=0
420 GOTO 500
430 LET V1=0
440 GOTO 540
450 LET P=0
460 GOTO 540
470 LET P1=0
480 GOTO 540
500 PRINT
505 PRINT "VALUES FOR VOLUMES A
RE IN MILLI-LITRES"
510 PRINT
515 PRINT "VALUES FOR PRESSURES
ARE IN CM OF MERCURY"
520 PRINT
525 PRINT
530 PRINT "TRY THESE VALUES FOR
V,V1,P,P1:"
540 PRINT "V:";V;TAB 8;"V1:";V1
;TAB 16;"P:";P;TAB 24;"P1:";P1
550 PRINT "WHAT IS YOUR APPROXI
MATE ANSWER USING BOYLES LAW"
560 INPUT Q
565 PRINT Q
570 PRINT "THE CORRECT ANSWER I
S---"
580 GOTO 150
```

```
                BOYLES LAW

INPUT YOUR VALUE FOR V,V1,P,P1
IN THAT ORDER.
FOR UNKNOWN VALUES USE 0 (ZERO).
V   0
V1  100
P   100
P1  300

ORIGINAL VOLUME=300



VALUES FOR VOLUMES ARE IN MILLI-
LITRES

VALUES FOR PRESSURES ARE IN CM
OF MERCURY


TRY THESE VALUES FOR V,V1,P,P1:
V:0      V1:5400 P:221    P1:245
WHAT IS YOUR APPROXIMATE ANSWER
USING BOYLES LAW
6000
THE CORRECT ANSWER IS---
ORIGINAL VOLUME=5986.4253



V:7300   V1:0     P:278    P1:165
WHAT IS YOUR APPROXIMATE ANSWER
USING BOYLES LAW
8600
THE CORRECT ANSWER IS---
NEW VOLUME=12299.394



V:5800   V1:6200 P:0      P1:169
WHAT IS YOUR APPROXIMATE ANSWER
USING BOYLES LAW
160
THE CORRECT ANSWER IS---
ORIGINAL PRESSURE=180.65517



V:5300   V1:7400 P:211    P1:0
WHAT IS YOUR APPROXIMATE ANSWER
USING BOYLES LAW
179
THE CORRECT ANSWER IS---
NEW PRESSURE=151.12162
```

Photoelectric Emissions

When light of a short wavelength falls on a metal surface, electrons are emitted from the metal. According to the description of this phenomenon by Einstein, there is a maximum wavelength for every metal above which no electrons are emitted. This is called the critical wavelength of the metal.

This program simulates a laboratory experiment in which a metal is placed in a vacuum and bombarded with soft X-rays. The number of electrons emitted is collected and measured with a microammeter. The program simulates three trials at each of nine wave lenghts.

After each set of experimental data, the program asks if you would like another run at a higher light intensity. The reason for doing this is that sometimes at low light intensities, not enough electrons have been emitted for meaningful measurements.

You can increase the precision of the experiment by increasing the number of wavelengths at which it is run. This value can be changed in Statement 500; note that the variable L is divided into 1000 in order to express the wavelength in Angstroms. One Angstrom (A) equals 10^{-8} centimeters or 10 microns.

Here are the coefficients for several metals:

Silver	.308
Bismuth	.338
Cadmium	.318
Lead	.340
Platinum	.385

It is a rare physics laboratory in a high school or college today that has the experimental apparatus to run this experiment, yet with a small computer the equipment can be simulated. There are many other things with which you would not normally be able to experiment that can be simulated with a computer. Things such as a nuclear power plant, a malaria epidemic, an urban mass transit system, and a bicycle factory.

Can you write a simulation for a real world system? You will find sections on simulations in the books *Computers in Mathematics* and *Computers in Science and Social Studies.* These, along with articles in *SYNC* and *Creative Computing* magazines, might be of some help in writing a simulation of your own.

```
 100 PRINT
 110 PRINT "ENTER METAL'S COEFFI
CIENT:";
 120 INPUT V
 130 PRINT V
 460 LET K=INT (1+2*RND)
 470 PRINT
 480 PRINT " WAVE ";MEASURED CURR
ENT-MIC AMPS"
 490 PRINT "LENGTH";TRIAL  1";TRIA
L 2";TRIAL 3"
 500 FOR L=.42 TO .25 STEP -.02
 510 LET M=INT (1000/L)
 520 PRINT M;
 530 FOR J=1 TO 3
 540 IF L>V THEN GOTO 570
 550 LET I=50R (INT (25*RND))
 560 GOTO 580
 570 LET I=50R (K*K*100+INT (35*
RND))
 580 LET N=INT (10*I+.5)/10
 590 PRINT TAB J*8;N;
 600 NEXT J
 610 PRINT
 620 NEXT L
 625 PRINT
 630 PRINT "INCREASE LIGHT INTEN
SITY (1-YES,2-NO)"
 660 INPUT Q
 670 IF Q=2 THEN STOP
 690 PRINT "BY WHAT FACTOR (1 TO
 10) ";
 700 INPUT F
 710 PRINT F
 720 LET K=K*F
 725 PAUSE 4E4
 727 CLS
 730 GOTO 470
```

```
ENTER METAL #5 COEFFICIENT:0.34

  WAVE ▌MEASURED CURRENT-MIC AMPS
LENGTH▌TRIAL   1▌TRIAL 2▌TRIAL 3
2380     20.3     20.4     20.1
2500     20.6     20.5     20
2631     20       20.6     20.3
2777     20.8     20.7     20.4
2941     2        4.7      2.4
3125     1.4      4.6      4.2
3333     1.7      4        3.7
3571     4.6      2.2      4.5
3846     2.6      4.6      2.6

INCREASE LIGHT INTENSITY (1-YES,
2-NO)
BY WHAT FACTOR (1 TO 10) 5


  WAVE ▌MEASURED CURRENT-MIC AMPS
LENGTH▌TRIAL   1▌TRIAL 2▌TRIAL 3
2380     100.2    100.2    100.1
2500     100      100.1    100
2631     100      100      100
2777     100      100      100.1
2941     2.2      1.7      3.2
3125     4.2      2        3.3
3333     4.2      3.3      4.6
3571     4.5      3.6      4.8
3846     4.6      3.7      4.7

INCREASE LIGHT INTENSITY (1-YES,
2-NO)
```

Mutation of Moths

This program is a simulation of the growth of a colony of pepper moths. The program allows a genetic mutation to be introduced in some year between 1 and 30. The mutation can favor either dark or light colored moths.

The program as it appears starts with a total colony of light moths; you could add an initial group of dark colored moths in Statement 255 as P1.

The sample run shows a mutation which favors dark moths occurring in Year 3. The mutation affects about 2% of the light moths each year and causes them to become dark. The program displays the number of moths of each color over a 30-year period.

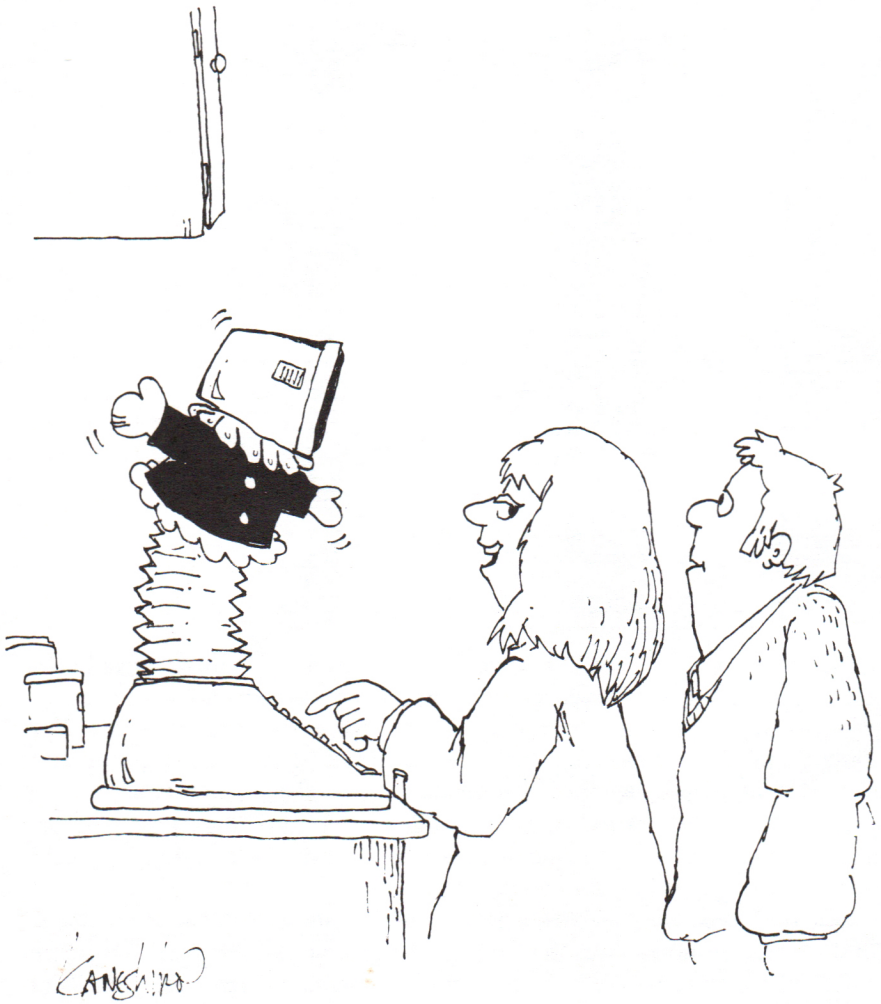
Obviously, because of the long time period involved it would be difficult to carry out this experiment in school, so the computer again is of real benefit.

```
 210 PRINT
 215 PRINT "ENTER RATE OF MUTATI
ON (1 TO 10)  ";
 220 INPUT M
 230 PRINT M
 250 LET P0=VAL "10000"
 260 LET Z=P0
 450 PRINT "YEAR IN WHICH ENVIRO
NMENTAL","CHANGE OCCURS?  ";
 455 INPUT X
 460 PRINT X
 464 LET L=1
 466 LET D=2
 470 PRINT "CHANGE FAVORS LIGHT
MOTHS OR","DARK (ENTER 1 OR 2)?
 ";
 480 INPUT E
 490 PRINT E
 580 CLS
 581 SCROLL
 582 PRINT "YEAR";TAB 8;"DARK MO
THS";TAB 21;"LIGHT MOTHS"
 610 FOR T=1 TO 30
 620 IF T>=X THEN GOTO 650
 630 LET P1=0
 640 GOTO 710
 650 IF E<>2 THEN GOTO 630
 660 LET P1=INT (P1+.01*M*P0+.5)
 670 LET P0=INT (Z-P1+.5)
 680 IF P1<Z THEN GOTO 710
 690 LET P1=Z
 700 LET P0=0
 710 SCROLL
 720 PRINT T;TAB 8;P1;TAB 21;P0
 730 NEXT T
```

```
ENTER RATE OF MUTATION (1 TO 10)
  2
YEAR IN WHICH ENVIRONMENTAL
CHANGE OCCURS?  3
CHANGE FAVORS LIGHT MOTHS OR
DARK (ENTER 1 OR 2)?  2

YEAR      DARK MOTHS      LIGHT MOTHS
1         0               10000
2         0               10000
3         200             9800
4         396             9604
5         588             9412
6         776             9224
7         960             9040
8         1141            8859
9         1318            8682
10        1492            8508
11        1662            8338
12        1829            8171
13        1992            8008
14        2152            7848
15        2309            7691
16        2463            7537
17        2614            7386
18        2762            7238
19        2907            7093
20        3049            6951
21        3188            6812
22        3324            6676
23        3458            6542
24        3589            6411
25        3717            6283
26        3843            6157
27        3966            6034
28        4087            5913
29        4205            5795
30        4321            5679
```

The so-called "killer bees" that were introduced into South America in the mid 70's were supposed to help honey production because they were much more energetic than the lazy honey bees in Brazil. The idea was that they would mate with the existing honey bees and produce a more productive strain. However, they went on a rampage killing people and terrorizing the country. Then they started to migrate north. U.S. agricultural officials became alarmed that they would invade this country and bring death and destruction.

Over the years, the killer bees have started to mate with the more docile South American honey bees, but only at the rate of 3% per year. Assuming they pose a danger to the U.S. until their numbers are reduced to 15% of their original quantity, how many years will it take for this to occur? The present migration patterns will bring them to the U.S. by 1989; will they still be dangerous (assume that the year of introduction was 1974)?



"I've just programmed our computer to give surprise birthday parties."

9

Potpourri

Here are four programs that didn't seem to fit anywhere else. The first is a game, although you may come to think of the lunar landing simulation as a game also. One is a nifty simulation of smog, and the other calculates depreciation by three different methods.

Number Guessing Game

Here is a computer program that plays the popular number guessing game. In it the computer picks a secret number between 1 and 100. You attempt to guess that number in as few tries as possible.

There are many ways to go about guessing the secret number. Let some of your friends play this game and see what approaches they use to find the secret number. One approach is to start with a guess of 10. If this is too low, increase each guess by 10 until the computer says that a guess is too high. When this point is reached, start from the previous guess and increase each guess by 1. This is the method used to solve some of the problems in the chapter on convergence.

Another approach is to try to bracket the number between upper and lower limits and reduce the limits by steps until the number is finally found. Two of the convergence programs used this approach.

Is one of these the best way? Well, these methods are not bad, particularly compared to starting with 1 and simply counting to 100 until the solution is found. But there is a better way. It is known as binary search.

This technique involves dividing the search domain, in this case 1 to 100, in half, and then in half again, and so on until the secret number is found. Play the game many times using different approaches. In the long run you should find that the binary search approach is the most efficient.

In Line 100, the program contains the statement that "you should not need more than 7 guesses." Why? If the secret number was between 1 and 128, what is the maximum number of guesses that would be necessary to find it? What if the number range were 1 to 130; then what would be the maximum number of guesses?

Revise the program to choose a secret number between 1 and 10,000. Now the upper limit is 100 times the 1 to 100 game here which requires a maximum of seven guesses to find the secret number; how many guesses will now be required?

Can you write a program in which the roles of the computer and player are reversed? In other words, the computer will try to guess your secret number between 1 and some upper limit. After each guess, you enter L for low, H for high, or C for correct. Can you write this program so the computer can tell if you are cheating, i.e., giving it inconsistent clues?

```
  10 PRINT "I HAVE CHOSEN AN INT
EGER 1","THROUGH 100.  TRY TO GU
ESS MY"
  15 PRINT "NUMBER IN AS FEW TRI
ES AS","POSSIBLE."
  16 PRINT
  20 LET C=0
  25 LET N=INT (RND*100)+1
  30 LET C=C+1
  31 IF PEEK 16442<12 THEN PAUSE
 300
  32 IF PEEK 16442<12 THEN CLS
  35 PRINT "YOUR GUESS IS?  ";
  40 INPUT G
  43 PRINT G
  45 IF N=G THEN GOTO 70
  50 IF G>N THEN GOTO 60
  55 PRINT "TOO LOW"
  56 GOTO 30
  60 PRINT "TOO HIGH"
  65 GOTO 30
  70 PRINT "CORRECT IN ";C;" GUE
SSES."
  75 IF C>3 THEN GOTO 90
  80 PRINT "YOU WERE LUCKY."
  85 STOP
  90 IF C>7 THEN GOTO 100
  95 PRINT "GOOD JOB."
  98 STOP
 100 PRINT "BUT YOU SHOULDNT NEE
D MORE THAN 7 GUESSES."
```

```
I HAVE CHOSEN AN INTEGER 1
THROUGH 100.  TRY TO GUESS MY
NUMBER IN AS FEW TRIES AS
POSSIBLE.

YOUR GUESS IS?  50
TOO HIGH
YOUR GUESS IS?  25
TOO LOW
YOUR GUESS IS?  37
TOO HIGH
YOUR GUESS IS?  32
CORRECT IN 4 GUESSES.
GOOD JOB.
```

```
I HAVE CHOSEN AN INTEGER 1
THROUGH 100.  TRY TO GUESS MY
NUMBER IN AS FEW TRIES AS
POSSIBLE.

YOUR GUESS IS?  50
TOO LOW
YOUR GUESS IS?  65
CORRECT IN 2 GUESSES.
YOU WERE LUCKY.
```

Depreciation—Three Methods

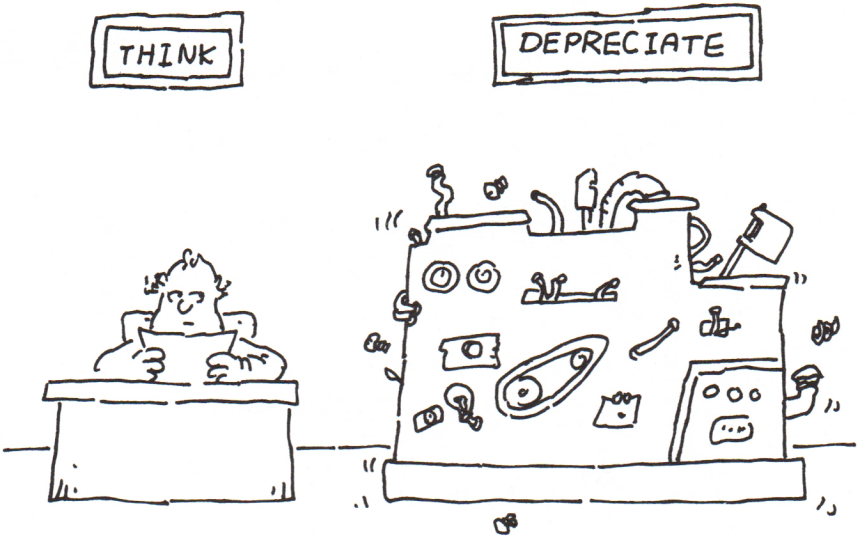
This program shows how a piece of capital equipment depreciates according to three commonly used methods of depreciation: straight line, sum of the year digits, and double declining.

The program asks for the original cost of the item, its expected life in years (the period of time over which it is to be depreciated), and its expected scrap (or sale) value at the end of that time. A table showing the annual depreciation for each of the three methods is then displayed.

```
120 PRINT "ORIGINAL COST?  ";
130 INPUT C
135 PRINT C
140 PRINT "  LIFE OF ITEM?  ";
150 INPUT L
155 PRINT L
160 PRINT "   SCRAP VALUE?  ";
170 INPUT S
175 PRINT S
180 PRINT
190 PRINT "YEAR  STR.    SUM OF
  DOUBLE"
200 PRINT TAB 6;"LINE";TAB 13;"
DIGITS    DECLINING"
210 PRINT
220 LET V=C-S
230 LET D1=V/L
240 LET Y=((L+1)/2)*L
250 LET Z=L
260 FOR X=1 TO L
270 LET D2=V*(Z/Y)
280 LET Z=Z-1
290 LET D3=2*C/L
300 LET C=C-D3
310 PRINT X;TAB 4;
312 LET Q=D1
314 GOSUB 400
316 LET Q=D2
317 PRINT TAB 13;
318 GOSUB 400
320 LET Q=D3
321 PRINT TAB 22;
322 GOSUB 400
325 PRINT
330 NEXT X
350 STOP
400 LET Q=INT (Q*100+.5)/100
420 IF Q>100 THEN GOTO 440
430 PRINT " ";
440 IF Q>10 THEN GOTO 460
450 PRINT " ";
460 PRINT "$";Q;
490 RETURN
```

```
ORIGINAL COST?   1000
 LIFE OF ITEM?   5
  SCRAP VALUE?   100

YEAR   STR.      SUM OF     DOUBLE
       LINE      DIGITS     DECLINING

1      $180       $300       $400
2      $180       $240       $240
3      $180       $180       $144
4      $180       $120        $86.4
5      $180        $60        $51.84
```

```
ORIGINAL COST?   2000
 LIFE OF ITEM?   4
  SCRAP VALUE?   300

YEAR   STR.      SUM OF     DOUBLE
       LINE      DIGITS     DECLINING

1      $425       $680       $1000
2      $425       $510       $500
3      $425       $340       $250
4      $425       $170       $125
```



Smog Simulation

This program is an adaptation of the smog model originally written by Herbert Peckham. The model assumes that vehicular traffic is the sole producer of smog, a somewhat poor assumption. It also assumes that most automobile traffic occurs during the daylight hours and that traffic volume is very low (actually, zero) at night. The smog generated by the cars is dissipated by atmospheric conditions which vary depending upon sunlight, temperature, and weather. All of these conditions may be specified by the user.

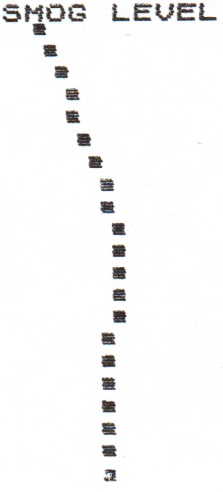
The model could be improved significantly by taking into account other sources of smog, by varying vehicular traffic according to the hour of the day, and by allowing daily variation of weather factors. Nevertheless, even in this rudimentary form it is interesting and instructive.

A plot of the smog level is produced in Statements 320 and 340. Under some conditions, the smog level reaches a value that cannot be plotted because it is greater than the width of the screen (and printer). For runs with conditions like this, you might want to delete the plotting routine. A more elegant solution would be to estimate the maximum value of the smog level from the input factors and calculate an appropriate plotting multiplication factor.

```
   5 LET D=0
  10 LET S=0
  15 LET T=7
  45 PRINT "CARS PER ROAD MILE--
"
  50 PRINT
  55 PRINT "LOS ANGELES   200 CAR
S PER MILE"
  60 PRINT "     DETROIT   100";TA
B 6;"TULSA   25"
  65 PRINT "HOW MANY CARS PER RO
AD MILE","IN SMOG CITY?  ";
  70 INPUT C
  73 PRINT C
  75 PAUSE 4E4
  77 CLS
  80 PRINT "SMOG GENERATION CONS
TANT"
  85 PRINT TAB 11;"1950 AUTO    ,
005";TAB 11;"1972 AUTO    .001";T
AB 11;"BUS          .010"
 100 PRINT "WHAT IS THE AVERAGE
VALUE IN    SMOG CITY?  ";
 105 INPUT L1
 106 PRINT L1
 108 PRINT
```

```
 110 PRINT "DAYTIME SMOG BREAKDO
WN RATE","IN PERCENT PER HOUR--"
 115 PRINT "  CLEAR  0.02";TAB 2
;"CLOUDY 0.01"
 120 PRINT "RATE IN SMOG CITY?
";
 125 INPUT R1
 130 PRINT R1
 135 PAUSE 4E4
 137 CLS
 140 PRINT "NIGHTTIME SMOG BREAK
DOWN RATE","IN PERCENT PER HOUR-
-"
 145 PRINT "  HOT   0.05";TAB 2;
"COOL  0.10"
 155 PRINT "RATE IN SMOG CITY?
";
 160 INPUT R2
 165 PRINT R2
 167 PRINT
 170 PRINT "SMOG DISPERSION FROM
 WEATHER","IN PERCENT PER HOUR--
"
 175 PRINT "  HIGH WIND AND RAIN
   1.00";TAB 2;"NO WIND AND DRY
    0.01"
 185 PRINT "DISPERSION FROM WEAT
HER IN SMOG CITY?  ";
 190 INPUT R3
 195 PRINT R3
 200 PAUSE 4E4
 205 CLS
 207 SCROLL
 210 PRINT "HOUR","SMOG LEVEL"
 215 SCROLL
 220 LET R=R1
 225 LET K1=L1
 230 LET T1=INT ((T-6)/12)
 240 IF T1/2=INT (T1/2) THEN GOT
O 270
 250 LET K=1
 260 LET R=R2
 270 LET S=S+K1*C*10-R*S-R3*S
 275 IF S<=0 THEN LET S=0
 280 LET T=T+1
 290 LET T3=INT (T/12)
 300 LET T2=T-T3*12+1
 310 LET S=INT (S*1000)/1000
 320 PRINT T2;TAB 3;"PM";TAB 7;S
 330 IF T3/2=IN( (T3/2) THEN PRI
NT AT 21,3;"A"
 335 IF D=0 THEN GOSUB 350
 337 LET D=1
 340 PLOT 35+ABS (M-S),1
 345 SCROLL
 350 GOTO 220
 360 LET M=S
 370 RETURN
```

```
CARS PER ROAD MILE--

LOS ANGELES   200 CARS PER MILE
    DETROIT   100
      TULSA   25
HOW MANY CARS PER ROAD MILE
IN SMOG CITY?   100

SMOG GENERATION CONSTANT
             1950 AUTO    .005
             1972 AUTO    .001
             BUS          .010
WHAT IS THE AVERAGE VALUE IN
SMOG CITY?   .001

DAYTIME SMOG BREAKDOWN RATE
IN PERCENT PER HOUR--
  CLEAR   0.02
  CLOUDY 0.01
RATE IN SMOG CITY?   .01

NIGHTTIME SMOG BREAKDOWN RATE
IN PERCENT PER HOUR--
  HOT    0.05
  COOL   0.10
RATE IN SMOG CITY?   0.1

SMOG DISPERSION FROM WEATHER
IN PERCENT PER HOUR--
  HIGH WIND AND RAIN    1.00
  NO WIND AND DRY       0.01
DISPERSION FROM WEATHER IN SMOG
CITY?   .05


HOUR                SMOG LEVEL
9   AM   0.999
10  AM   1.939
11  AM   2.822
12  AM   3.652
1   PM   4.432
2   PM   5.166
3   PM   5.856
4   PM   6.504
5   PM   7.113
6   PM   7.686
7   PM   8.224
8   PM   7.99
9   PM   7.791
10  PM   7.622
11  PM   7.478
12  PM   7.356
1   AM   7.252
2   AM   7.164
3   AM   7.089
4   AM   7.025
5   PM   6.971
```



Lunar Lander Simulation

This program is one of the most popular computer simulations around. It is available in many versions; this one is adapted from the original program written in 1969.

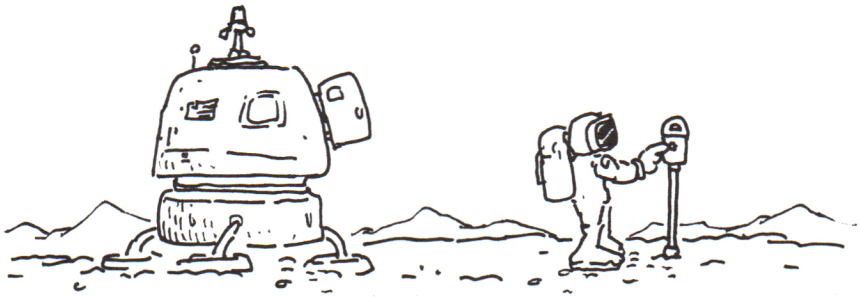
The program represents an exact simulation of an Apollo lunar landing module during the final descent. This portion of the descent would normally be controlled by the on-board computer backed up by another computer in the lunar orbiter, and still another computer on Earth. However, to exercise your knowledge of physics and to make an interesting game, we will assume that all three computers have had a simultaneous malfunction. Hence, it is up to you to land the spacecraft safely.

To make a soft landing, you may change the burn rate of the retro rockets every ten seconds. You have a choice of not firing at all (burn rate of 0) or of firing at a fuel rate of between 8 and 200 pounds per second. Engine ignition occurs at 8 pounds, hence values between 1 and 7 pounds are not possible. You have 16,500 pounds of fuel. This is 500 pounds more than an actual LEM has, which will give you a little margin for error. When you get proficient, change Statement 35 to N = 16000 to simulate the real thing more closely. The capsule weight is 33,000 pounds.

Not that this is the way to come in, but if you did not fire the rockets at all, the estimated time for a free fall descent is 120 seconds to impact (and a huge splat).

This program did not have room for one interesting calculation that is in the original. In particular, if you crash, the original program tells you the depth of the new lunar crater that you created. But you can figure this out for yourself; the depth in feet is equal to 0.277 times the landing speed in miles per hour.

Good luck!

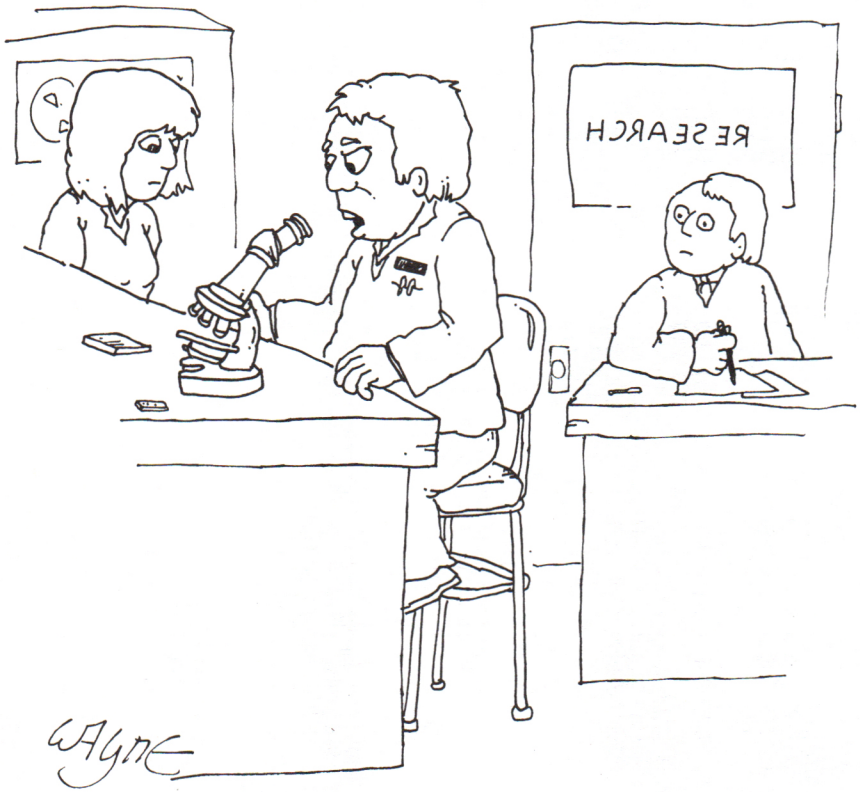


```
  1 SCROLL
  5 LET L=0
 10 PRINT "SEC   MI+FT       MPH
 FUEL     BURN"
 15 SCROLL
 16 PRINT
 20 LET A=120
 25 LET V=1
 30 LET M=33000
 35 LET N=16500
 40 LET G=.001
 45 LET Z=1.8
 50 LET P=INT (3600*V+.5)
 51 IF P<1 THEN LET P=INT ((360
0*V+.5)*1000)/1000
 52 SCROLL
 54 PRINT L;TAB 5;INT A;" ";INT
 (5280*(A-INT A));TAB 15;P;TAB 2
1;M-N;TAB 28;
 55 INPUT K
 60 PRINT K
 65 LET T=10
 70 IF M-N<.001 THEN GOTO 120
 72 IF T<.001 THEN GOTO 50
 75 LET S=T
 80 IF M>=N+S*K THEN GOTO 90
 85 LET S=(M-N)/K
 90 GOSUB 315
 95 IF I<=0 THEN GOTO 245
100 IF V<=0 THEN GOTO 110
105 IF J<0 THEN GOTO 275
110 GOSUB 215
115 GOTO 70
120 SCROLL
122 PRINT "RAN OUT OF FUEL AT "
;L;" SEC"
125 LET S=(-V+SQR (V*V+2*A*G))/
G
130 LET V=V+G*S
135 LET L=L+S
140 LET W=3600*V
142 SCROLL
145 PRINT "ON THE MOON AT ";INT
L;" SEC--IMPACT"
146 SCROLL
148 PRINT "VELOCITY ";W;" MPH"
150 SCROLL
152 IF W>1.2 THEN GOTO 165
155 PRINT "PERFECT LANDING"
160 STOP
165 IF W>10 THEN GOTO 180
170 PRINT "GOOD LANDING"
175 STOP
180 IF W>60 THEN GOTO 200
185 PRINT "CRAFT DAMAGE.  YOU A
RE STRANDED."
190 STOP
200 PRINT "YOU BLEW IT.  NO SUR
VIVORS."
205 STOP
```

```
 2215  LET L=L+S
 2220  LET T=T-S
 2225  LET M=M-S*K
 2230  LET A=I
 2235  LET V=J
 2240  RETURN
 2245  IF S<.005 THEN GOTO 140
 2250  LET D=V+SQR (V*V+2*A*(G-Z*K
/M))
 2255  LET S=2*A/D
 2260  GOSUB 315
 2265  GOSUB 215
 2270  GOTO 245
 2275  LET W=(1-M*G/(Z*K))/2
 2280  LET S=M*V/(Z*K*(W+SQR (W*W+
V/Z)))+.05
 2285  GOSUB 315
 2290  IF I<=0 THEN GOTO 245
 2295  GOSUB 215
 2300  IF J>0 THEN GOTO 70
 2305  IF V>0 THEN GOTO 275
 2310  GOTO 70
 2315  LET Q=S*K/M
 2320  LET J=V+G*S-Z*Q*(1+Q*(.5+Q*
(1/3+Q*(.25+Q/5))))
 2325  LET I=A-G*S*S/2-V*S+Z*S*Q*(
.5+Q*(1/6+Q*(1/12+Q/20)))
 2330  RETURN
```

SEC	MI+FT	MPH	FUEL	BURN
0	120 0	3600	16500	0
10	109 5015	3636	16500	0
20	99 4223	3672	16500	0
30	89 2903	3708	16500	0
40	79 1055	3744	16500	0
50	68 3960	3780	16500	0
60	58 1055	3816	16500	0
70	47 2904	3852	16500	200
80	37 1884	3483	14500	200
90	28 1191	3087	12500	200
100	20 1251	2660	10500	200
110	13 2549	2197	8500	200
120	8 370	1693	6500	200
130	4 658	1139	4500	200
140	1 4204	527	2500	100
150	0 4042	212	1500	45
160	0 1864	84	1050	20
170	0 908	46	850	17
180	0 438	18	680	13
190	0 269	5	550	10
200	0 213	3	450	10
210	0 189	0.925	350	9.2
220	0 179	1	258	9.3
230	0 165	1	165	7.1
240	0 91	9	94	4

```
ON THE MOON AT 244 SEC--IMPACT
VELOCITY 18.398722 MPH
CRAFT DAMAGE.  YOU ARE STRANDED.
```



"Rats! A bacterium just ate the new micro-mini computer."

References

Magazines referred to in the text include:

- *SYNC.* This magazine focuses on Timex and Sinçlair computers exclusively. It carries articles and reviews of peripherals, software and other accessories.
- *Creative Computing.* This is the leading magazine of software and applications for all small computers. It carries articles, tutorials, how-to applications, and extensive in-depth evaluations.

Books referred to in the text include:

- *Computers in Mathematics: A Sourcebook of Ideas.* Hundreds of classroom-tested ideas for using computers to learn about mathematics.
- *Computers in Science and Social Studies.* Scores of simulation programs in biology, ecology, physics and management of real world systems.

Other books of interest to Timex and Sinclair computer owners.

- *The Best of SYNC.* Compilation of the best articles from *SYNC magazine.*
- *The Gateway Guide to the ZX81.* Material for the novice user.
- *Getting Acquainted with your ZX81.* Practical applications; includes 80 ready-to-run programs.
- *The ZX81 Companion.* For the more advanced user; contains material on real time graphics, information processing, educational applications and a listing of the monitor.
- *Computers for Kids — Sinclair Edition.* Programming primer for children from age 8 on.

All of these books and magazines are available from *Creative Computing.* Write or call for the current price:

Creative Computing
39 E. Hanover Ave.
Morris Plains, NJ 07950
(800) 631-8112
In NJ (201) 540-0445

THE TIMEX-SINCLAIR 1000 $8.95

IDEA BOOK

This **Ideabook** contains dozens of ways to make the most out of your computer for solving practical, everyday problems. The 50 ready-to-run programs demonstrate scores of different techniques for solving problems in mathematics, science and business.

The ten chapters deal with solving problems by formulas and repetitive trials, convergence, recursion, compounding, probability, geometry, science, simulations, and drill and practice.

Some of the problems demonstrate the capabilities of the computer; others identify its shortcomings. It is important to be familiar with both the strengths and weaknesses of your tools so you can recognize the types of jobs for which they are suitable.

The author, David H. Ahl, has been involved with the use of computers since 1957. He is the author of 16 books and is the founder and editorial director of Creative Computing, SYNC and Video & Arcade Games magazines.

CREATIVE COMPUTING PRESS

ISBN # 0-916688-48-